REFINED ISOGEOMETRIC ANALYSIS: A SOLVER-BASED DISCRETIZATION METHOD

Daniel Garcia

Supervised by *David Pardo* and *Victor M. Calo*

May 29, 2018

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

REFINED ISOGEOMETRIC ANALYSIS: A SOLVER-BASED DISCRETIZATION METHOD

Daniel Garcia

Supervised by *David Pardo* and *Victor M. Calo*

May 29, 2018

This dissertation have been possible with the support of the Project of the Spanish Ministry of Economy and Competitiveness with reference MTM2016-76329-R (AEI/FEDER, EU), and MTM2016-81697-ERC/AEI, the BCAM "Severo Ochoa" accreditation of excellence SEV-2013-0323, and the Basque Government through the BERC 2014-2017 program and the Consolidated Research Group Grant IT649-13 on "Mathematical Modeling, Simulation, and Industrial Applications (M2SI)"; the European Union's Horizon 2020 research and innovation program under the Marie Sklodowska-Curie grant agreement No 644602, and the Texas Advanced Computing Center (TACC) at The University of Texas at Austin which provides the HPC resources that contributed to the research results reported within this Dissertation.

Acknowledgements

Above all, I am eternally grateful to Professor David Pardo for all the support that he gave me, even before I initiated my Ph.D. I greatly appreciate the unequaled advices that he has given to me during my Ph.D., which have allowed me to grow both personally and professionally. I am confident to say that Professor David Pardo is an admirable advisor and that his guidance was what made possible the development of this high-quality scientific work.

I also thank Professor Victor Calo for his supervision during the development of this Dissertation and for the friendly invitations to visit him in Saudi Arabia, and later in Australia. I enjoyed working with him as well as sharing with his family and members of the research group the Argentinian barbecue times. I am very grateful to him for his help and advice that allowed me to begin with the doctoral studies.

I would like to give special thanks to Lisandro Dalcin, researcher scientist at King Abdullah University of Science & Technology (KAUST) in Saudi Arabia, who has given me his unconditional support throughout the development of the scientific work presented in this document. His experience and intuition in codding topics have helped me in the development of my research implementation as well as improving my coding struggles.

I wish to thank all the colleagues from the Mathematical Modeling, Simulation, and Industrial Applications (M^2SI) group for their friendship, support, and help. In particular, Dr. Théophile Chaumont-Frelet for the constructive discussions of many details of this scientific work and his useful advice, as well as Dr. Vincent Darrigrand for the debates and teachings on LaTeX that allowed me to create high-quality documents for scientific distribution. To both of them, I am also grateful for their effort in helping me to expand my knowledge in modern languages, "Merci beaucoup". Besides, I thank Adel Sarmiento, former Ph.D. student at KAUST, who helped me with the main pillars to apply my research to the study of incompressible fluid flows.

I would like to thank my parents and brother. They have given me their unconditional support throughout all my journey, and they deserve more than a mere expression of appreciation.

Abstract

Isogeometric Analysis (IGA) is a computational approach frequently employed nowadays to study problems governed by Partial Differential Equations (PDEs). This approach defines the geometry using conventional Computer Aided Design (CAD) functions and, in particular, Non-uniform Rational Basis Splines (NURBS). These functions represent complex geometries commonly found in engineering design and are capable of preserving exactly the geometry description under refinement as required in the analysis. Moreover, the use of NURBS as basis functions is compatible with the isoparametric concept, allowing to build algebraic systems directly from the computational domain representation based on spline functions, which arise from CAD. Therefore, it avoids to define a second space for the numerical analysis resulting in huge reductions in the total analysis time.

To perform the numerical analysis, we can use either direct or iterative solvers. Direct solvers are preferred to study stiff linear problems that are not solvable with iterative solvers, as well as another type of problems, e.g., problems with multiple Right Hand Side (RHS). Moreover, when the problem size and the FLoating Point Operations (FLOPs) required to solve the problem are enormous, and the direct solvers become excessively expensive, the iterative solver turns into a more suitable alternative.

For the case of direct solvers, the performance strongly depends upon the employed discretization method. In particular, on IGA, the continuity of the solution spaces plays a significant role in their performance. High continuous spaces degrade the direct solver's performance, increasing the solution times by a factor up to $\mathcal{O}(p^3)$ with respect to traditional Finite Element Analysis (FEA) per unknown, being p the polynomial order.

In this work, we propose a solver-based discretization that employs highly continuous finite element spaces interconnected with low continuity hyperplanes to maximize the performance of direct solvers. Starting from a highly continuous IGA discretization, we introduce C^0 hyperplanes, which act as separators for the direct solver, to reduce the interconnection between Degrees of Freedom (DoF) in the mesh. By doing so, both the solution time and best approximation errors are simultaneously improved. We call the resulting method "refined Isogeometric Analysis (rIGA)".

While this method can be applied to a variety of problems, in this Dissertation we focus on analyzing the impact of the continuity reduction when solving a Laplace problem with structured meshes and uniform polynomial orders in both 2D and 3D. Numerical results indicate that rIGA delivers speed-up factors proportional to p^2. For instance, in a 2D mesh with four million elements and $p = 5$, the linear system resulting from rIGA is solved 22 times faster than the one from highly continuous IGA. In a 3D mesh with one million elements and $p = 3$, the linear rIGA system is solved 15 times faster than the IGA one.

We then develop a version of the rIGA strategy that introduces hyperplanes of arbitrary continuity (C^k hyperplanes with $0 \leqslant k \leqslant p-1$). This strategy, called Optimally refined Isogeometric

Abstract

Analysis (OrIGA), leads to more efficient discretization than those obtained with the original version of rIGA. By using separators of arbitrary continuity degree, we achieve a performance boost of up to 25% in the direct solvers with respect to rIGA. Thus, the savings with respect to IGA are larger than in rIGA.

We have also designed and implemented a similar rIGA strategy for iterative solvers. This strategy splits the mesh into subdomains using C^0-hyperplanes and constructs the Schur complements of the subdomains (macro-elements) using a direct solver. We then assemble those Schur complements into a global skeleton system, which is only composed of the DoF located along the boundaries of all the subdomains. Subsequently, we solve this system iteratively using Conjugate Gradients (CG) with an Incomplete LU (ILU) preconditioner. Lastly, a backward substitution is performed to recover the eliminated DoF and obtain the solution of the original system. Thus, rIGA for iterative solvers is a hybrid solver strategy that combines a direct solver (static condensation step) to eliminate the internal macro-elements DoF, with an iterative method to solve the skeleton system.

The hybrid solver strategy achieves moderate savings with respect to IGA when solving a 2D Poisson problem with a structured mesh and a uniform polynomial degree of approximation. For instance, for a mesh with four million elements and polynomial degree $p = 3$, the iterative solver is approximately 2.6 times faster (in time) when applied to the rIGA system than to the IGA one. These savings occur because the skeleton rIGA system contains fewer non-zero entries than the IGA one. The opposite situation occurs for 3D problems, and as a result, 3D rIGA discretizations provide no gains with respect to their IGA counterparts.

In this work, we also apply rIGA to solve incompressible fluid flow problems on an enclosed domain. To satisfy the inf-sup stability condition and guarantee divergence-free discrete solutions, the implemented rIGA employs a combination of C^0 and C^1 hyperplanes to reduce the continuity on the solution spaces. Numerical results show that the L^2 norm of the discretization error improves as we reduce the continuity. Therefore, rIGA delivers smaller errors than C^{p-1} IGA discretizations. In terms of the computational savings, rIGA provides a reduction in the computational cost of the direct solvers by a factor of $\mathcal{O}(p^2)$ in both 2D and 3D.

Resumen

En la actualidad, diferentes áreas de la ingeniería se centran en el estudio de procesos físicos gobernados por Ecuaciones en Derivadas Parciales (EDPs). Algunos de estos procesos son la mecánica de sólidos, la dinámica de fluidos, la termodinámica y el electromagnetismo. Las aplicaciones prácticas de estos procesos físicos producen complejos sistemas matemáticos, lo que dificulta o incluso imposibilita el encontrar una solución analítica (exacta). Sin embargo, es posible aproximar la solución empleando métodos numéricos.

A lo largo de las últimas décadas se han desarrollado una gran variedad de técnicas numéricas, incluidas las diferencias finitas, los volúmenes finitos y los elementos finitos. La mayoría de los métodos numéricos se pueden utilizar para resolver cualquiera de estos problemas. No obstante, para aproximar la solución de un problema particular es habitual utilizar el método que mejor se ajuste al mismo. El método de los elementos finitos (MEF), por ejemplo, es una excelente herramienta para estudiar una amplia variedad de problemas en diferentes áreas de la ingeniería y es frecuentemente usado para realizar análisis estructurales ya que es altamente flexible para analizar diseños de ingeniería en diferentes entornos [78, 124, 125, 50, 2, 1, 72, 37, 63, 126].

En esta tesis, nos centraremos en los **métodos de Galerkin**, entre los que se encuentra el MEF. Para resolver numéricamente un problema gobernado por EDPs que cuenta con condiciones de borde específicas usando los métodos de Galerkin, se requiere primero proporcionar una representación geométrica del dominio del problema. El diseño asistido por ordenador (CAD, *Computer Aided Design* por sus siglas en inglés) es una técnica estándar utilizada hoy en día para representar computacionalmente estas geometrías. Actualmente, las herramientas basadas en CAD utilizan varios tipos de funciones, por ejemplo B-splines y/o NURBS, lo que permite realizar representaciones geométricas de alta calidad de los dominios físicos. Después de construir la representación computacional, se procede a generar la representación discreta del dominio computacional, conocida como malla. Por medio de dicha malla y usando una formulación variacional de las EDPs que consiste en utilizar funciones de base definidas en el espacio de discretización, construimos el sistema de ecuaciones algebraicas que corresponde a la representación discreta del problema numérico. En el MEF tradicional, las funciones de base restringidas a un elemento (denominadas funciones de forma) se definen en un elemento de referencia, y se emplea una biyección entre el elemento de referencia y el elemento en el dominio físico [78].

En algunos casos, la generación de la malla require más recursos computacionales que la posterior construcción del sistema algebraico y solución del mismo. Los requerimientos computacionales para generar la representación discreta del dominio aumentan con el tamaño y la complejidad del mismo. Además, mallas finas y bien estructuradas son necesarias para reducir el error de discretización resultante de la incapacidad de las mallas de elementos finitos para capturar imperfecciones en la geometría. Varias aplicaciones en sectores industriales como el sector biomédico, automotriz y aeronáutico han mostrado que el análisis numérico dedica

aproximadamente el 80 % del tiempo computacional total para construir las mallas (discretizar el dominio) [79].

El **análisis Isogeométrico** (IGA, *isogeometric analysis* por sus siglas en inglés) es un método de Galerkin desarrollado para resolver problemas gobernados por EDPs. Este método fue introducido por primera vez en el año 2005 por Hughes et al. [79] (véase también [43]). En IGA, la representación geométrica se construye usando funciones CAD convencionales (que en general son funciones NURBS o B-Splines). Estas funciones permiten representar geometrías complejas comúnmente encontradas en problemas de ingeniería. Asimismo, estas funciones son capaces de preservar exactamente la descripción de la geometría bajo refinamiento como se requiere en el análisis de problemas. Además, el uso de NURBS como funciones base es compatible con el concepto isoparamétrico, es decir, el mismo conjunto de funciones de base se puede utilizar para la representación computacional de la geometría y el análisis del problema. A pesar de las limitaciones que existen en softwares actuales, la capacidad de usar directamente las funciones CAD para definir el espacio de IGA evita la necesidad de definir un segundo conjunto de funciones (así como las proyecciones correspondientes) para realizar el análisis numérico [79]. En última instancia, IGA permite genera el sistema algebraico directamente desde la representación del dominio. Por lo tanto, este enfoque no sólo logra una reducción importante del coste total del tiempo de análisis, sino que también elimina las imprecisiones debidas a las imperfecciones de la reproducción de la geometría en la malla.

IGA ha sido empleado en muchos campos de ingeniería desde que fue concebido en [79]. Este método ha mostrado algunas ventajas sobre los enfoques tradicionales. Las discretizaciones construidas con IGA usan funciones CAD altamente continuas. Estos tipos de funciones presentan ciertas ventajas en el análisis de la dinámica de fluidos [14, 19, 71, 29, 34, 3, 12, 60, 95, 97, 114, 18, 123] así como en el análisis de interacción fluido-estructura [16, 15, 82]. Además, IGA permite construir fácilmente discretizaciones de alto orden de continuidad, lo que lo convierte en un método atractivo para resolver problemas que requieren derivadas de alto orden, como por ejemplo, fenómenos de transición de fase (gobernados por la ecuación de Cahn-Hilliard), gradientes de deformación (elasticidad) [61], y problemas de difusión de fracturas [115].

Adicionalmente, IGA ha mostrado ciertas ventajas con respecto a los métodos tradicionales para problemas en cáscaras y placas [20, 6, 85, 21, 47, 22, 88]. En particular, las funciones CAD de alta continuidad permiten construir representaciones exactas de la placa y de las cáscaras. Con IGA es posible construir exitosamente formulaciones para resolver problemas con grandes deformaciones y libres de rotaciones. Además, la alta continuidad en IGA permite predecir mejor la deformación de objetos delgados donde las mallas de elementos finitos presentan elementos altamente deformados y donde el MEF tiene problemas para predecir la cinemática de deformación [20].

En problemas de elasticidad en régimen lineal, vibración estructural, y propagación de ondas, las funciones de alta continuidad usadas en IGA proporcionan robustez y precisión [79, 45, 7, 87]. Para los problemas de vibración estructural, el método de refinamiento en k, un concepto de refinamiento único de IGA, entrega espectros de frecuencia más precisos que cuando se usan elementos finitos de alto orden (refinamiento en p). En particular, a altas frecuencias IGA aproxima mejor los autovalores del sistema que con los métodos tradicionales de elementos finitos.

IGA también se ha utilizado para resolver problemas de optimización [120, 91, 106, 105] y mecánica de sólidos [44, 45, 7, 87, 26], al igual que para estudiar problemas en los campos de las ciencias médicas [123, 17, 77, 32] y electromagnetismo [30]. En esos casos, IGA ha mostrado que puede proporcionar discretizaciones que alcanzan el mismo nivel de error que un MEF tradicional pero con un número menor de grados de libertad [79, 44, 13].

Los sistemas algebraicos obtenidos al implementar cualquiera de los métodos de discretización mencionados anteriormente pueden resolverse usando tanto métodos directos como iterativos. Los métodos directos se usan a menudo para resolver problemas con sistemas lineales rígidos, los cuales no pueden ser resueltos usando métodos iterativos debido a que estos no convergen o los resultados no son confiables. Otros escenarios en los que los resolvedores directos son convenientes incluyen problemas con sistemas que cuentan con múltiples lados de la derecha, por ejemplo, aquellos asociados a problemas inversos. Además, los resolvedores directos son el bloque principal de muchos resolvedores iterativos [41]. En la tesis, nos centraremos primero en un resolvedor directo multifrontal, el cual es un resolvedor directo de vanguardia. Este resolvedor directo fue propuesto originalmente en [53].

Los **resolvedores directos** basados en el método de particionamiento de grafos, como el resolvedor directo multifrontal, dividen recursivamente el grafo de conectividad de la malla del sistema en pares de subdominios interconectados por pequeños subconjuntos de incógnitas llamados separadores. El orden de eliminación de las incógnitas es establecido por la estructura recursiva del grafo particionado, eliminando primero los grados de libertad asociados a los subdominios, y luego se prosigue eliminando aquellos que se encuentran asociados a los separadores que conectan los subdominios.

El costo de resolver un sistema algebraico de ecuaciones, y en específico, el costo de factorizar la matriz usando el método de factorización LU, depende del método de discretización usado. Los estudios presentados en [38, 33] muestran que en IGA, la continuidad de las funciones de base juega un papel fundamental en la degradación del rendimiento del resolvedor directo por incógnita. En particular, con una discretización de IGA donde la continuidad es máxima (C^{p-1}), el resolvedor directo es $\mathcal{O}(p^3)$ veces más costoso por incógnita que si usamos un MEF tradicional, siendo p el grado polinomial del sistema. En estos análisis se asumió una continuidad uniforme y un número total de incógnitas fijo en el sistema.

El rendimiento de los resolvedores directos por grado de libertad mejora cuando reducimos la continuidad en la interfaz entre algunos de los elementos que conforman la malla para un problema C^{p-1} dado. Sin embargo, si en una malla con un número de elementos fijo reducimos la continuidad global hasta C^0, el costo total de la solución del sistema puede llegar a ser mayor que el del sistema original C^{p-1}. El crecimiento en el costo se debe al aumento en el número de incógnitas que conlleva la reducción de la continuidad.

En este trabajo, **nosotros proponemos** un método de discretización basado en el resolvedor directo. Este método emplea espacios de elementos finitos altamente continuos interconectados por medio de hiperplanos de baja continuidad para maximizar el rendimiento de los resolvedores directos. Sobre una discretización altamente continua (C^{p-1}) introducimos hiperplanos C^0, que actúan como separadores para el resolvedor directo. Estos hiperplanos reducen la interconexión entre las funciones base en la malla, lo que resulta en una reducción del tiempo de solución y una mejora en el error de aproximación. Esto último ocurre debido a que la reducción local de

la continuidad enriquece el espacio de elementos finitos. Llamamos al método de discretización resultante, **análisis isogeométrico refinado** (rIGA, *refined IsoGeometric Analysis* por sus siglas en inglés).

Si bien este método puede aplicarse a una variedad de problemas, en esta tesis doctoral nos centramos en analizar el impacto de la reducción de la continuidad en el rendimiento del resolvedor directo cuando solucionamos un problema de Laplace con mallas estructuradas y órdenes polinomiales uniformes tanto en 2D como en 3D. Las estimaciones teóricas del número de operaciones de coma flotante (FLOPs, *FLOating Point operations* por sus siglas en inglés) demuestran que nuestro método puede reducir el tiempo computacional total en un factor de entre p^2 y p^3. Los resultados numéricos avalan que rIGA proporciona factores de reducción proporcionales a p^2. Por ejemplo, en una malla bidimensional ($2D$) con cuatro millones de elementos y $p = 5$, resolvemos el sistema lineal obtenido con rIGA 22 veces más rápido que el obtenido usando IGA con funciones de base de alta continuidad (C^{p-1}). Además, en una malla tridimensional ($3D$) con un millón de elementos y $p = 3$, el sistema lineal obtenido con rIGA se resuelve 15 veces más rápido que cuando usamos IGA con funciones base de máxima continuidad (C^{p-1}).

Asimismo, hemos desarrollado una versión de rIGA que permite usar hiperplanos de continuidad arbitraria (hiperplanos C^k con $0 \leqslant k \leqslant p-1$). Esta versión optimizada de rIGA conduce a una discretización más eficiente que la obtenida con la versión original. Utilizando separadores con grado de continuidad arbitrario, alcanzamos un aumento del rendimiento del resolvedor directo de hasta un 25 $\%$ con respecto a rIGA. Por lo tanto, los ahorros en costos computacionales con respecto a IGA son mayores que en el caso de rIGA.

Los resolvedores directos se vuelven extremadamente caros e incluso imposibles de usar cuando intentamos resolver EDPs usando grandes sistemas dispersos de ecuaciones lineales con millones de incógnitas. Para resolver problemas de este tipo se requiere una cantidad considerable de memoria física y un enorme número de FLOPs. Una alternativa para resolver este tipo de problemas consiste en usar métodos iterativos. Este tipo de resolvedores requieren una cantidad de memoria de $\mathcal{O}(N)$, siendo N el número de incógnitas, permitiendo resolver problemas enormes. Los métodos iterativos resuelven los sistemas algebraicos (en su forma matricial $A\mathbf{x} = \mathbf{b}$) mejorando secuencialmente una estimación o conjetura inicial de la solución del problema ($\mathbf{x} = \mathbf{x}_0$). Estos métodos realizan una secuencia de productos matriz-vector ($A \cdot \mathbf{x}$) hasta que se satisface un criterio de parada, que típicamente consiste en alcanzar un valor de error (o residuo) que está por debajo de cierta tolerancia tol, es decir, $||A\mathbf{x} - b|| \leqslant tol$.

El costo de solucionar sistemas dispersos cuando se usan resolvedores iterativos depende del número de operaciones requeridas por el método iterativo (básicamente productos matriz-vector) y del costo de construcción y aplicación del precondicionador [99]. En trabajos anteriores se muestra que el coste de los resolvedores iterativos por incógnita aumenta cuando se usan discretizaciones de Galerkin con funciones de base altamente continuas como es el caso de IGA [39]. En particular, un producto matriz-vector es $\mathcal{O}(p^2)$ veces más costoso cuando usamos una discretización con continuidad máxima (C^{p-1}) que para el caso del MEF tradicional con condensación estática. Asimismo, el coste de construcción y aplicación del precondicionador depende del método de discretización. Por ejemplo, la técnica de precondicionamiento *elemento por elemento* [80] muestra un factor de incremento de $\mathcal{O}(p^3)$ en el costo cuando se usa IGA con funciones base de alta continuidad (C^{p-1}) en lugar de un MEF. Una de las opciones más baratas

para precondicionar el sistema cuando utilizamos IGA como método de discretización consiste en una factorización LU incompleta [99].

En esta tesis doctoral, también hemos diseñado e implementado un método basado en rIGA para **resolvedores iterativos**. Este método divide la malla en subdominios usando hiperplanos que reducen la continuidad hasta C^0 y calcula los complementos de Schur de los subdominios (macro-elementos) usando un resolvedor directo. A continuación, ensambla los complementos de Schur en un sistema reducido global, también denominado sistema del esqueleto. Este sistema se compone de las incógnitas situadas a lo largo de las interfaces entre los subdominios. Posteriormente, se resuelve el sistema reducido de forma iterativa utilizando el método del gradiente conjugado (GC) con un precondicionador basado en la factorización LU incompleta. Por último, se realiza una sustitución hacia atrás para recuperar las incógnitas eliminadas y obtener la solución del sistema original. Por lo tanto, la discretización basada en rIGA para métodos iterativos es un resolvedor híbrido que combina un método directo (condensación estática) para eliminar las incógnitas interiores en los macro-elementos, con un método iterativo para resolver el sistema reducido.

El resolvedor híbrido ha mostrado ser más económico que el resolvedor iterativo basado en GC y precondicionado con el método de factorización LU incompleta para resolver el problema de Poisson en 2D con una malla estructurada y un orden polinomial de aproximación uniforme. Para una malla con cuatro millones de elementos y un grado polinomial $p = 3$, la discretización obtenida con rIGA se resuelve 2.6 veces más rápido usando el resolvedor híbrido que para la obtenida con IGA. El ahorro en costos computacionales se produce porque el sistema reducido obtenido con rIGA contiene menos entradas no nulas que la matriz original obtenida con IGA. La situación opuesta ocurre para problemas en 3D, y como resultado, las discretizaciones en 3D obtenidas con rIGA no proporcionan ganancias con respecto a sus contrapartes IGA y MEF.

Por último, hemos aplicado el método de discretización rIGA para resolver **problemas de mecánica de fluidos** considerando un fluido incompresible en un dominio cerrado. Para satisfacer la condición de estabilidad inf-sup y garantizar soluciones discretas del campo de velocidades con divergencia cero puntual, la discretización obtenida con rIGA emplea una combinación de hiperplanos C^0 y C^1 para reducir la continuidad de las funciones de base. Los resultados numéricos muestran que la norma L^2 del error de discretización mejora a medida que se reduce la continuidad. Por lo tanto, rIGA entrega soluciones con errores más pequeños que los obtenidos con IGA y máxima continuidad (C^{p-1}). En términos de ahorro computacional, **rIGA proporciona una reducción en el costo de los resolvedores directos en un factor de $\mathcal{O}(p^2)$ tanto en 2D como en 3D.**

Contents

List of Figures

Acronyms

PDEs	Partial Differential Equations
CAD	Computer Aided Design
FDM	Finite Difference Methods
FVM	Finite Volume Methods
FEA	Finite Element Analysis
IGA	Isogeometric Analysis
rIGA	refined Isogeometric Analysis
OrIGA	Optimally refined Isogeometric Analysis
BCs	Boundary conditions
DoF	Degrees of Freedom
NZ	NonZero
RHS	Right Hand Side
FLOPs	FLoating Point Operations
NURBS	Non-uniform Rational Basis Splines
ILU	Incomplete LU
EBE	Element-by-Element
CG	Conjugate Gradients
GMRES	Generalized Minimal RESiduals
BiCG	Biconjugate Gradients
BiCGSTAB	Biconjugate Gradients stabilized
NKS	Newton-Krylov-Schwarz
JNFK	Jacobian-free Newton-Krylov
BDDC	Balancing Domain Decomposition by Constraints
FSI	Fluid Structure Interaction

1. Introduction

1.1. Motivation

Nowadays, many engineering fields focus on studying physical processes governed by Partial Differential Equations (PDEs). For instance, those governed by solid mechanics, fluid dynamics, thermodynamics, electromagnetics and so on. The practical applications of those processes involve complex PDEs systems making difficult or even impossible to find an analytic (exact) solution. A suitable alternative consists of approximating the solution of those problems by employing numerical methods. A large variety of numerical techniques has been developed during the last decades such as Finite Difference Methods (FDM), Finite Volume Methods (FVM), and Finite Element Analysis (FEA). In practice, most of the numerical methods are suitable for solving any class of multiphysics problem. However, we use the method that better fits the requirements of a problem to approximate its solution. For instance, a numerical method as FEA, which is an excellent tool to study a broad variety of engineering problems in different fields, is often used for solving structural analysis due to its flexibility in analyzing an engineering design (domain) in almost any possible environment [78, 124, 125, 50, 2, 1, 72, 37, 63, 126]. In this thesis, we will focus on FEA.

To solve numerically a problem governed by PDEs and specific Boundary conditions (BCs) using FEA, we first provide a geometrical representation of the problem domain. Computer Aided Design (CAD) is a standard technique used nowadays to build these computational representations. Moreover, CAD tools currently use B-Splines type functions which allow performing high-quality representations of the physical domains. After constructing the computational representation with CAD, we generate a discrete representation of the computational domain known as mesh. Based on this mesh and using a variational formulation of the governing PDEs with trial and test functions defined by their respective FEA basis functions, we construct the system of algebraic equations which corresponds to the discrete representation of the numerical problem. In traditional FEA, the basis functions are defined on a reference element, and a mapping to the physical element is employed [78]. Figure 1.1 sketches the computational domain representation and mesh used to perform a structural analysis (stress analysis) of a propeller using a discretization approach based on FEA.

It is sometimes the case that the mesh generation requires more computational resources than the subsequent construction and solution of the algebraic system. The computational requirement for the mesh generation enlarges as the size and complexity of the domain increases. This occurs because a finer and better-structured mesh is needed to lessen the discretization error resulting from the inability of FEA mesh to capture some geometry imperfections. In many applications in industrial sectors such as biomedicine, automotive and aeronautics, they show that the numerical analysis dedicates approximately 80 percent of the total computational time

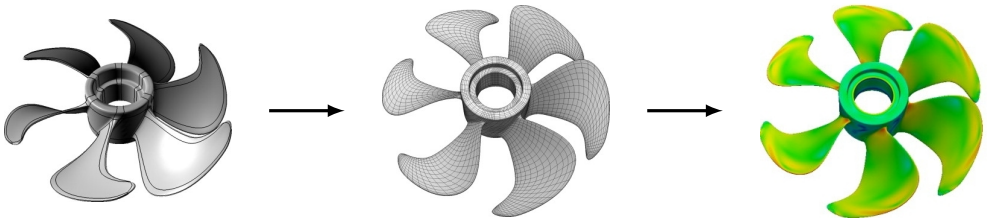


Figure 1.1.: Geometry, Mesh and stress contour (structural analysis) of a propeller. This structural analysis was developed in [96] by M. A. Scott et al.

to build the meshes (discretize the CAD domain) [79].

Isogeometric Analysis (IGA) is a well-established computational approach to solve problems governed by PDEs. The concept of IGA was first introduced in 2005 by Hughes et al. [79], see also [43]. IGA defines the geometry using conventional CAD functions and, in particular, Non-uniform Rational Basis Splines (NURBS). These functions represent complex geometries commonly found in engineering design and are capable of preserving exactly the geometry description under refinement as required in the analysis. Moreover, the use of NURBS as basis functions is compatible with the isoparametric concept, that is, the same set of basis functions can be used for both geometry representation and analysis. Although many obstacles remain in the present-day CAD software, this feature avoids the need to define a second set of functions (and the corresponding transfer operators) for the numerical analysis [79]. Ultimately, IGA allows to generate the algebraic system directly from the CAD representation of the domain. Thus, this approach not only achieves a huge reduction of the total analysis time cost but also removes the inaccuracies due to mesh imperfections. Chapter 2 explains the fundaments of IGA that are relevant to the thesis.

Since the main idea was established in [79], IGA has been employed in many engineering fields in which it has shown some advantages over traditional approaches. IGA discrete representations use highly continuous CAD basis functions (e.g., NURBS). The smoothness of those basis functions often benefits the analysis of fluid dynamics [14, 19, 71, 29, 34, 3, 12, 60, 95, 97, 114, 18, 123] and fluid-structure iteration problems [16, 15, 82]. Moreover, IGA permits to easily construct high order discretizations becoming attractive for solving problems that require high order derivatives of the variable of study. For instance, to approximate the solution of phase transition phenomena (based on Cahn-Hilliard equation) [70, 119, 116, 118, 117], gradient elasticiy [61] and diffuse fracture processes [115] problems.

IGA has also shown benefits over traditional approaches for solving problems on shells and plates [20, 6, 85, 21, 47, 22, 88]. In particular, the smooth basis functions allow to construct exact representations of plate and shells. Moreover, IGA has great success in constructing large deformation and rotation-free formulations. Additionally, IGA discretizations exhibit much less shear-locking compared with FEA [20].

In the case of linear elasticity, structural vibration, and wave propagation, the smooth IGA

basis functions provide additional robustness and accuracy [79, 45, 7, 87]. For structural vibration problems, the k-method, a refinement concept unique from IGA, provides more accurate frequency spectra than when using high order Finite Elements p-refinements. In particular, IGA discretizations eliminate the optical branches, which are the cause of severe accurate degradation in higher modes when using high order p-refinements on FEA.

IGA has also been used in the context of optimisation problems [120, 91, 106, 105], solid mechanics [44, 45, 7, 87, 26], medical applications [123, 17, 77, 32] and electromagnetics [30]. In those, IGA shows that it can provide discretizations with a lower number of Degrees of Freedom (DoF) than those delivered by traditional FEA and achieves the same error level [79, 44, 13].

Either direct or iterative solvers can be used to compute the solution of the algebraic systems resulting from the methods mentioned above. Direct solvers are often used to solve stiff linear problems where iterative solvers do not converge or are unreliable. Other scenarios in which direct solvers are convenient include problems with multiple right-hand sides (e.g., when solving inverse problems). Moreover, direct solvers are the main building blocks of many iterative solvers [41]. In the thesis, we will focus on a multifrontal solver, which is the state-of-the-art direct solver and was originally proposed in [53].

Direct solvers based on graph partitioning, as the multifrontal direct solver, recursively split the system's connectivity graph into pairs of subdomains interconnected by small subsets of DoF called separators. The order of elimination of the DoF is set by the recursive structure of the partitioned graph, eliminating first the DoF associated with the subdomains, and then those associated to the separators that connect the subdomains. A detailed explanation is provided in Chapter 3.

The cost to solve an algebraic system of equations, specifically the cost to perform the LU factorization of the matrix, is determined by the discretization method. Previous works presented in [38, 33] show that in IGA, the continuity plays a significant part in the degradation of the direct solver's performance on a per DoF. In particular, a maximal continuity IGA discretization is $\mathcal{O}(p^3)$ times more expensive than traditional FEA per unknown, with p being the polynomial order. This analysis assumes uniform continuity and a fixed total number of DoF in the system. Table 1.1 illustrates the computational cost resulting from using highly continuous basis functions (IGA) as well as traditional FEA.

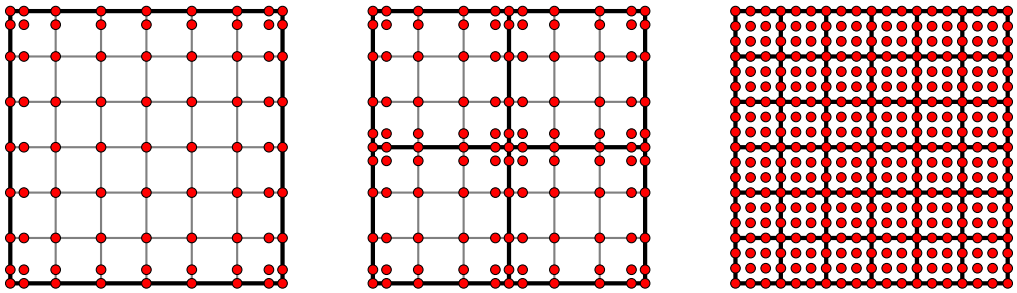
System continuity	FLOPS	
	Skeleton	Static condensation
Traditional FEA $\left(C^0\right)$	$\mathcal{O}\left(\left(N^{(d-1)/d}\right)^3\right)$	$\mathcal{O}\left(Np^{2d}\right)$
Maximal continuity IGA $\left(C^{p-1}\right)$	$\mathcal{O}\left(\left(N^{(d-1)/d}p\right)^3\right)$	$\mathcal{O}(1)$

N = degrees of freedom, p = polynomial order, d = dimension (2 or 3)

Table 1.1.: Summary of Floating Point Operations (FLOPs) estimates derived in [38, 33].

The performance of direct solvers per unknown improves when reducing the inter-element

continuity for a given C^{p-1} problem (see Table 1.1). However, if the number of elements is kept fixed and the global continuity is turned to be C^0, then the total solution cost of the system may become larger than that of the original C^{p-1} system. That growth in the cost is due to the increased number of DoF (N) that the reduction of continuity carries (see Figure 1.2).



(a) No continuity reduction (C^{p-1} IGA). (b) Partial continuity reduction. (c) Full continuity reduction (C^0 FEA).

Figure 1.2.: Illustration of Galerkin discretizations of a 2D system composed of 6×6 elements with polynomial basis functions of order $p = 3$. Red circles represent the nodal degrees of freedom in the system, while black lines denote the mesh skeleton. Bold lines represent lower continuity.

Direct solvers become prohibitively expensive or even impossible to use when the computation of the solution of the PDEs involves large sparse systems (of the order of millions). In those cases, a considerable amount of memory and a large number of FLOPs are required to solve the problems. Iterative solvers are a more suitable alternative in these situations. These methods require an amount of memory of $\mathcal{O}(N)$, allowing to solve huge problems. Iterative methods solve the algebraic system (in its matrix form $A\mathbf{x} = \mathbf{b}$) by sequentially improving an initial guess of the problem solution $\mathbf{x} = \mathbf{x}_0$. Essentially, the method performs a sequence of matrix-vector products ($A \cdot \mathbf{x}$) until a stopping criterion is satisfied, which typically corresponds when reaching an error (or residual) value that is below a certain tolerance tol, i.e., $||A\mathbf{x} - b|| \leqslant tol$.

The cost to solve sparse systems when using iterative solvers depends on the number of operations required by the iterative method (basically matrix-vector products) and the cost of setup and applying the preconditioner [99]. Previous works presented in [39] show that the cost of iterative solvers increases when using Galerkin-based discretizations with highly continuous basis functions as IGA. In particular, a matrix-vector product operation is $\mathcal{O}(p^2)$ times more expensive for maximal continuity IGA discretization than for traditional FEA. This cost reduces to approximately a factor of eight when static condensation is not employed. Moreover, the cost of setup and application of the preconditioner also depends on the discretization approach. For instance, the Element-by-Element (EBE) preconditioning technique shows an increment by a factor of $\mathcal{O}(p^3)$ in the cost when using on C^{p-1} IGA rather than on C^0 FEA, or the Incomplete LU (ILU) factorization preconditioning method that becomes up to a constant more expensive in same circumstances. Ultimately, ILU factorization technique is one of the cheapest preconditioning options when using smooth basis functions [99].

1.2. Main contribution

In this work, we analyze the impact of various continuity patterns in the computational cost of both direct and iterative solvers for a fixed mesh topology and polynomial order p. The strategy we propose introduces some hyperplanes that partition the domain into subdomains, besides reducing the continuity along the hyperplanes location. As the continuity is reduced, the interconnection among the subdomains weakens, and the number of degrees of freedom in the system grows. This is equivalent to discretizing the system using a variation of the traditional FEA that employs C^{p-1} subsystems (subdomains) as elements (macro-elements). In the limiting case of reducing continuity along all the separators, the resulting system becomes a traditional C^0 FEA. Alternatively, the method can also be interpreted as a high continuity IGA with certain refinements over specific hyperplanes that locally reduce the continuity. Figure 1.2 illustrates the cases with no reduction of continuity (C^{p-1} IGA) and full reduction of continuity (C^0 FEA), in addition to a third case (center) with partial reduction of continuity that consists of a C^0 skeleton and four macro-elements (C^{p-1} subsystems).

The main contribution of this dissertation is to present an approach that delivers a class of discretizations which have minimal solution cost. We call the approach refined Isogeometric Analysis (rIGA). The corresponding discretization spaces obtained with rIGA are finer than standard maximal continuity IGA spaces and are faster to solve for than both traditional FEA and IGA for meshes with a fixed number of elements. Moreover, the enrichment of the discrete space results in an increment of the accuracy of the best approximation error with respect to the C^{p-1} IGA discretization. This not always imply a direct improvement of the results. For example, stability problems in the solution may lead to a worse result even if the best approximation error is reduced.

In the dissertation, we combine the results from our publications [67, 65, 66]. In [67], we described the impact of continuity reduction in the solution cost when solving a Laplace example with a direct solver. We assumed that the inter-subdomains continuity is C^0, while the inter-element continuity inside the subdomains is C^{p-1}. By doing so, the cost of the LU factorization dramatically decreases. For instance, by reducing the continuity over certain hyperplanes on a third order 3D IGA discretization with two million elements, we reduce the solution time to approximately one hour, while the IGA discretization requires 15 hours to be computed (despite being coarser), and the corresponding FEA discretization needs over 100 hours.

In [65], we extended the approach allowing to reduce the continuity to arbitrary degrees. In this case, the inter-subdomain continuity is C^k, being $0 \leqslant k < p$. This variation leads to discretizations that can be more efficiently solved via direct solvers than the ones in the previous implementation. By allowing the use of arbitrary continuity degrees, it is possible to reduce the total computational time by a factor of approximately 60 when compared to IGA and FEA. This corresponds to a boost of 25% in the direct solver performance with respect to the case studied in [67].

Finally, in [66], we extended the analysis of continuity reduction to the case of iterative solvers and analyzed its main features and limitations. This extension involves an iterative solver that consists of four steps. First, the solver partitions the domain problem into subdomains. We refer to these as blocks or macro-elements. The mesh partitioning uses hyperplanes that reduce the

continuity over the inter-subdomain boundaries. Then, the solver performs a static condensation of the macro-elements internal DoF. This reduces the system size to just the DoF located along the macro-elements boundaries (mesh skeleton). Third, the iterative solver computes the solution of the reduced system. Finally, a backward substitution using the factors obtained when performing the static condensation allows us to recover the solution of the original system. Thus, our hybrid solver combines a direct solver to build the Schur complements of the macro-elements with an iterative solver to solve the skeleton system. This approach showed a moderate reduction in computational cost for 2D implementations, while in 3D no gains are observed.

1.3. Outline

The remainder of the dissertation is organized as follows: Chapter 2 recalls the key aspects of IGA, particularly the ones that are relevant to the scope of the thesis. Chapter 3 describes the numerical solvers used to analyze the impact of local continuity reduction on the computational cost. Chapter 4 formulates the new discretization strategy employed with direct and iterative solvers. Chapter 5 describes extensive numerical experimentations for a Laplace problem in 2D and 3D computed with a direct solver, and a Poisson problem in 2D and 3D solved iteratively. Chapter 6 analyses the numerical results of two fluid flow test problem when imposing continuity reduction on the discretization and are solved with a direct solver. The main conclusions and future works are stated in Chapter 7 and the main achievements in Chapter 8. This dissertation also contains an appendix A describing a formulation to estimate the memory required to solve a problem with a direct solver using rIGA.

2. Isogeometric Analysis

Isogeometric Analysis (IGA) employs Computer Aided Design (CAD) functions and, in particular, Non-uniform Rational Basis Splines (NURBS) as basis functions [79, 44, 43]. These functions represent complex geometries commonly found in engineering design and are capable of preserving exactly the geometry description under refinement as required in the analysis. Moreover, the use of NURBS as basis functions is compatible with the isoparametric concept, that is, the same set of basis functions can be used for both geometry representation and analysis. Also, highly continuous NURBS often provide better approximation properties than traditional Finite Element Analysis (FEA) on a per degree of freedom basis [96, 57].

In this chapter, we recall the fundamentals of IGA that are relevant for the scope of this thesis. We first introduce the B-splines basis functions. Next, we introduce the NURBS basis functions [103]. Lastly, we expose the IGA discretization.

2.1. B-Splines

The B-spline parametric space in IGA is the set of all possible combinations of one-dimensional coordinates arranged on knot vectors. These knot vectors are defined as

$$\Xi = (\xi_1, ..., \xi_{n_{bf}+p+1}) \qquad \forall\, \xi_i \in \mathbb{R}, \tag{2.1}$$

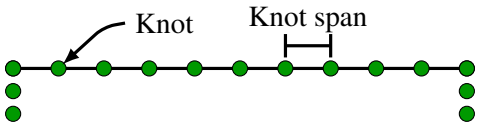
where ξ_i is the i-th knot, n_{bf} is the number of basis functions, and p is the polynomial order. The number of basis function is $n_{bf} = n_{\text{elem}} + p$, being $n_{\text{elem}} = \sqrt[d]{N_{\text{elem}}}$ the number of elements in one spatial dimension. The knot vector is *uniform* when the knots are equidistant to the surrounded knots, that is, when the intervals between two consecutive knots (knot spans) are equal. In case that at least one knot is not equally-spaced respect to a surrounding knot, the knot vector is *non-uniform*. Moreover, the knot vectors may involve knots located at the same coordinates (empty knot span), these are called *repeated* knots. An *open knot vector* is the knot vector that has $p+1$ repeated knots at both the first and the last coordinate. The *open knot vector* is the standard knot vector used in CAD (Figure 2.1).

The B-spline basis functions with support on the knot vector are defined by the Cox-de Boor recursion formula [46, 49]. The piecewise basis functions ($p = 0$) are defined by

$$\mathcal{N}_{i,0}(\xi) = \begin{cases} 1 & for \xi_i \leqslant \xi < \xi_{i+1}, \\ 0 & else, \end{cases} \tag{2.2}$$

while for linear ($p = 1$), cuadratic ($p = 2$), cubic ($p = 3$), and so on, the B-spline basis functions are defined as

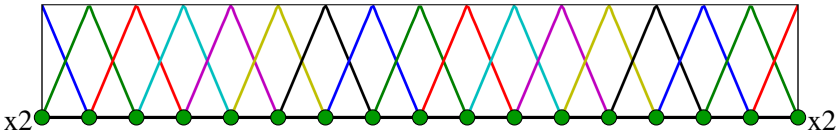
$$\mathcal{N}_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i}\mathcal{N}_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}}\mathcal{N}_{i+1,p-1}(\xi). \tag{2.3}$$



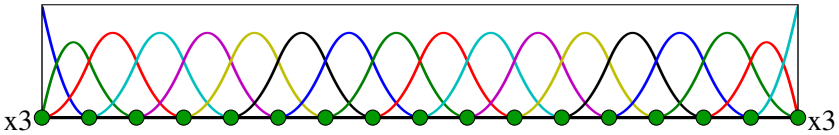
$$\Xi = (0, 0, 0, \tfrac{1}{10}, \tfrac{2}{10}, \tfrac{3}{10}, \tfrac{4}{10}, \tfrac{5}{10}, \tfrac{6}{10}, \tfrac{7}{10}, \tfrac{8}{10}, \tfrac{9}{10}, 1, 1, 1)$$

Figure 2.1.: Illustration of a *open knot vector* with 12 basis functions and polynomial degree $p = 2$.

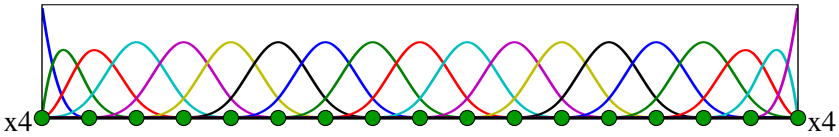
The resulting basis functions for polynomial degree $p = 0$ and $p = 1$ are the same as the traditional piecewise constant and linear finite element basis functions, respectively (Figure 2.2a and 2.2b). But the B-spline basis functions with higher polynomial degree ($p > 1$) are different from their FEA counterpart (Figure 2.2c). The shape of these functions depends on the location along the knot vector. In particular, the support of the basis functions is equal to the knot spans of $p + 1$ knots. For instance, the basis functions at the edges of an *open knot vector* have a support equal to $[\xi_0, \xi_{p+1}]$ and $[\xi_{n_{bf}}, \xi_{n_{bf}+p+1}]$, that results only in the knot spans $[\xi_p, \xi_{p+1}]$ and $[\xi_{n_{bf}}, \xi_{n_{bf}+1}]$ since the remaining knots involve empty knot spans. The internal basis functions involve a support equal to $[\xi_i, \xi_{i+p+1}]$ which is the sum of the knot spans between the $p + 1$ knots (see Figure 2.3a).



(a) Polynomial order $p = 1$



(b) Polynomial order $p = 2$



(c) Polynomial order $p = 3$

Figure 2.2.: Sets of B-spline basis functions associated to a one dimensional knot vector.

Moreover, the derivatives of the B-splines basis functions with respect to the spatial coordi-

nates are given by

$$\frac{\partial^m}{\partial^m \xi}\mathcal{N}_{i,p}(\xi) = \frac{p!}{(p-m)!}\sum_{j=0}^{m}\beta_{m,j}\mathcal{N}_{i+j,p-m}(\xi), \tag{2.4}$$

with

$$\begin{aligned}
\beta_{0,0} &= 1, \\
\beta_{m,0} &= \frac{\beta_{m-1,0}}{\xi_{i+p-m+1}-\xi_i}, \\
\beta_{m,j} &= \frac{\beta_{m-1,j}-\beta_{m-1,j-1}}{\xi_{i+p+j-m+1}-\xi_{i+j}} \quad j = 1, ..., m-1, \\
\beta_{m,m} &= \frac{-\beta_{m-1,m-1}}{\xi_{i+p+1}-\xi_{i+m}}.
\end{aligned}$$

Some important properties of the B-spline basis functions are:

- The support of each basis function is contained in $p+1$ knot spans (Figure 2.3a),

$$\text{Support: } \mathcal{N}_{i,p} = [\xi_i, \xi_{i+p+1}].$$

- The basis functions form a partition of unity $\forall\ \xi_i$ (Figure 2.3b),

$$\sum_{i=1}^{n_{bf}}\mathcal{N}_{i,p}(\xi) = 1.$$

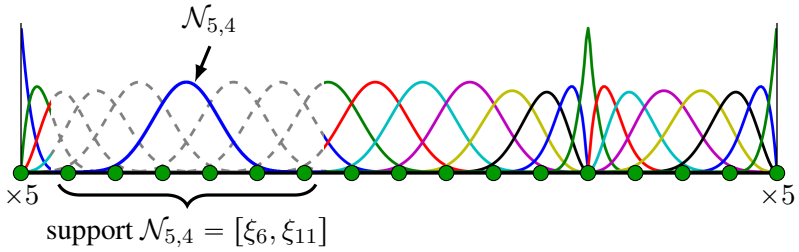
- The basis functions are piecewise polynomials of degree p and continuity C^{p-1}, except in the presence of *repeated* knots where the continuity reduces according to the number of knot repetitions. For instances if the knot is repeated r times, the continuity of the B-splines basis function that crosses this knot is C^{p-r} (Figure 2.3c).

- The first and the last basis functions on an *open knot vector* are interpolatory, while the remaining basis functions are zero at this location (edges of the knot vector). This also occurs on spatial coordinates with p *repeated* knots (where C^0 continuity is imposed) (Figure 2.3d).

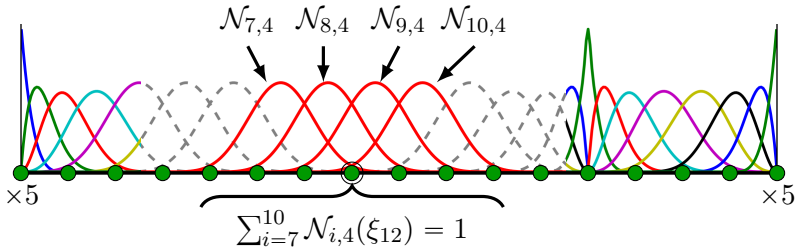
- The basis functions are non-negative over the entire knot vector (Figure 2.3),

$$\mathcal{N}_{i,p}(\xi) \geqslant 0 \quad \forall i = 1, ..., n_{bf}+p+1.$$

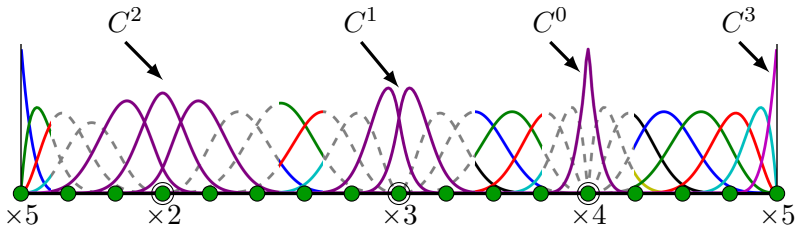
There are three ways of refining the parametric spaces spanned by the B-splines basis functions. These refining options consist of modifying the knot vector by knot insertion, degree elevation or k-refinement [44, 79, 43, 103]. The knot insertion is an analog of the h-refinement in traditional FEA. In this case, we enrich the knot vector by inserting knots at different locations from the original knots contained by the knot vector. Therefore, the inserted knot involve



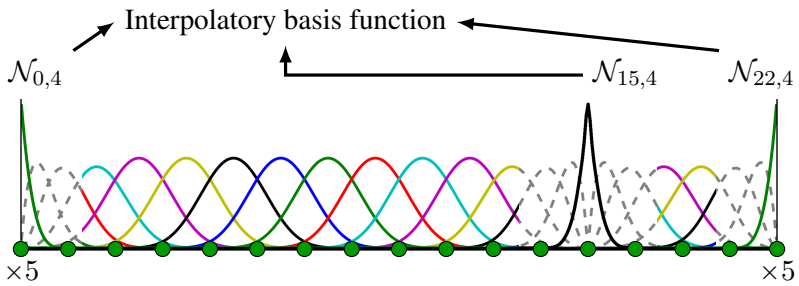
(a) Basis function support



(b) Partition of unity



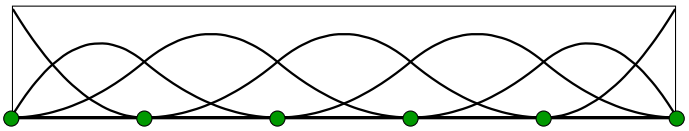
(c) *Repeated* knots



(d) Interpolatory basis functions

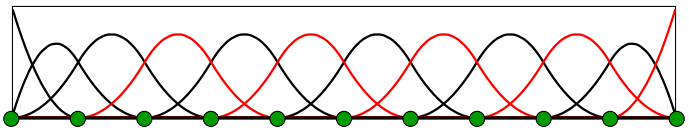
Figure 2.3.: Illustration of some important properties of B-spline basis functions.

non-empty knot spans. Figure 2.4b illustrates an example of knot insertion in which we insert five knots on the knot vector presented in Figure 2.4a. This reduces the knot spans size in half and increments the number of basis functions from 7 to 12.



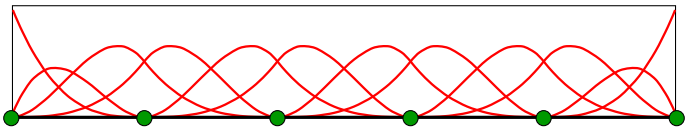
$\Xi = (0, 0, 0, \frac{2}{10}, \frac{4}{10}, \frac{6}{10}, \frac{8}{10}, 1, 1, 1)$

(a) Original knot vector with quadratic basis functions



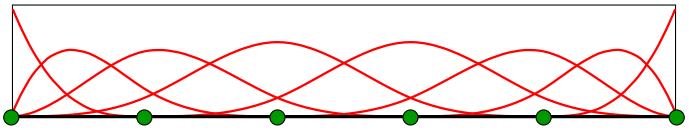
$\Xi = (0, 0, 0, \frac{1}{10}, \frac{2}{10}, \frac{3}{10}, \frac{4}{10}, \frac{5}{10}, \frac{6}{10}, \frac{7}{10}, \frac{8}{10}, \frac{9}{10}, 1, 1, 1)$

(b) Knot insertion.



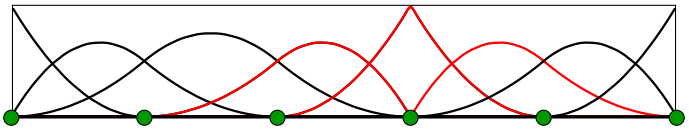
$\Xi = (0, 0, 0, 0, \frac{2}{10}, \frac{2}{10}, \frac{4}{10}, \frac{4}{10}, \frac{6}{10}, \frac{6}{10}, \frac{8}{10}, \frac{8}{10}, 1, 1, 1, 1)$

(c) Polynomial degree elevation



$\Xi = (0, 0, 0, 0, \frac{2}{10}, \frac{4}{10}, \frac{6}{10}, \frac{8}{10}, 1, 1, 1, 1)$

(d) pk-elevation



$\Xi = (0, 0, 0, \frac{2}{10}, \frac{4}{10}, \frac{6}{10}, \frac{6}{10}, \frac{8}{10}, 1, 1, 1)$

(e) k-refinement

Figure 2.4.: Ways of refining the parametric spaces spanned by the B-splines basis functions.

The second refining option is degree elevation. This refinement increases the polynomial degree of the B-spline basis functions, and at the same time increases the number of knot repetitions to preserve the continuity at the knots locations. This refinement concept is the analog of p-refinement for traditional FEA. Figure 2.4c illustrates a p-refinement for the knot vector presented in Figure 2.4a. In the example, we increase the polynomial degree from $p = 2$ to $p = 3$ while keeping the original continuity all along the knot vector. Thus, we increase the repetitions of all knots, which results in an increment of the number of basis functions from 7 to

12.

The third refinement concept was introduced with the name of k-refinement by Hughes et al. in [79]. However, we prefer to call this refinement ***pk*-elevation** since this consists of increasing both the polynomial degree and the smoothness (continuity) of the B-spline basis functions. To increase the continuity of the basis functions, we only need to keep the same number of repetitions of the original knots. The pk-elevation delivers spaces that contain r more basis functions, being r the number of degree elevation. For instance, the number of basis functions on a knot vector increases by one when the polynomial degree goes from $p = 2$ to $p = 3$ (Figure 2.4d). The pk-elevation is performed by increasing the number of repetition of the knots at the beginning and the end of the knot vector. This refinement concept does not have an analogous for FEA.

In this work, we denote k-refinement to the process that performs continuity reduction of the B-splines basis functions. This k-refinement concept enriches the spaces by inserting knots at the same original knot coordinates of the knot vector. Thus, increasing the number of knots repetitions. Figure 2.4e illustrates an example of k-refinement, in which we increase the repetition of one knot in the knot vector. This results in a reduction of the continuity degree at the knot location, increasing the number of basis functions from 7 to 8. In this work, we refer to our concept of k-refinement as k-reduction to avoid confusion, since the k-refinement title is already widely accepted in IGA to refer to pk-elevation.

2.1.1. B-spline geometric entities

We construct curves, surfaces, and solids in $\mathbb{R}$ by using sets of B-spline basis functions contained in knot vectors. The curves are composed of a linear combination of the B-splines basis functions on a single knot vector. The coefficients accompanying the basis functions are known as control points and correspond to the node values in traditional FEA. A curve constructed with a set of B-splines is given by

$$\mathcal{C}(\xi) = \sum_{i=1}^{n_{bf}} \mathcal{N}_{i,p}(\xi)\mathbf{B}_i, \tag{2.5}$$

where $\mathbf{B}_i \in \mathbb{R}^d$ is the i-th control point that corresponds to the coefficient of $\mathcal{N}_{i,p}$ basis function.

The surfaces are constructed by tensor product of two sets of B-splines basis functions. Then, given two knot vectors $\Xi = (\xi_i, ..., \xi_{n_{bf}+p+1})$ and $\mathscr{H} = (\psi_i, ..., \psi_{m_{bf}+\rho+1})$, as well as a set of control points arranged in a matrix form $B_{i,j}$, the surface is defined as

$$\mathcal{S}(\xi, \psi)^{p,\rho} = \sum_{i=1}^{n_{bf}} \sum_{j=1}^{m_{bf}} \mathcal{N}_{i,p}(\xi)\mathcal{M}_{j,\rho}(\psi)B_{i,j}, \tag{2.6}$$

where $\mathcal{N}_{i,p_\Xi}$ and $\mathcal{N}_{i,p_\mathscr{H}}$ are B-splines basis functions. Moreover, n_{bf} and m_{bf} are the number of basis function on Ξ and $\mathscr{H}$ knot vectors, respectively. p and ρ are the polynomial degrees imposed on each knot vector.

We construct the solids by a tensor product of three sets of B-splines basis functions. Disposing of $\Xi = (\xi_i, ..., \xi_{n_{bf}+p+1})$, $\mathscr{H} = (\psi_i, ..., \psi_{m_{bf}+\rho+1})$ and $\mathscr{L} = (\varphi_i, ..., \varphi_{l_{bf}+\varrho+1})$ knot

vectors, and the controls points ordered in tensor $B_{i,j,k}$, the solid is defined as

$$\mathcal{S}(\xi,\psi,\varphi)^{p,\rho,\varrho} = \sum_{i=1}^{n_{bf}} \sum_{j=1}^{m_{bf}} \sum_{k=1}^{l_{bf}} \mathcal{N}_{i,p}(\xi)\mathcal{M}_{j,\rho}(\psi)\mathcal{L}_{j,\varrho}(\varphi)B_{i,j,k}, \tag{2.7}$$

where $\mathcal{N}_{i,p_\Xi}$, $\mathcal{N}_{i,p_\mathscr{H}}$ and $\mathcal{L}_{i,p_\mathscr{L}}$ are the B-splines basis functions associated to each knot vectors, n_{bf}, m_{bf} and l_{bf} are the number of basis function on each knot vector, and p, ρ and ϱ are the polynomial degrees of each knot vector.

2.2. NURBS

The B-splines allow to represent a great variety of shapes in $\mathbb{R}$ accurately. However, in particular cases such as circles and ellipses, these functions deliver poor approximations. The non-uniform rational B-splines are a general version of the B-splines functions that permit the exact representation of different geometric entities including those resulting from conic sections (circles and ellipses). The NURBS basis functions in one dimension are defined as

$$\mathcal{R}_i^p(\xi) = \frac{\mathcal{N}_{i,p}(\xi)w_i}{\sum_{\hat{i}=1}^{n_{bf}} \mathcal{N}_{\hat{i},p}(\xi)w_{\hat{i}}}, \tag{2.8}$$

where w_i is a positive *weight* of the $\mathcal{N}_{i,p}$ basis function. The NURBS basis functions in 2D and 3D are given by

$$\mathcal{R}_{i,j}^{p,\rho}(\xi,\psi) = \frac{\mathcal{N}_{i,p}(\xi)\mathcal{M}_{j,\rho}(\psi)w_{i,j}}{\sum_{\hat{i}=1}^{n_{bf}}\sum_{\hat{j}=1}^{m_{bf}} \mathcal{N}_{\hat{i},p}(\xi)\,\mathcal{M}_{\hat{j},\rho}(\psi)w_{\hat{i},\hat{j}}}, \tag{2.9}$$

$$\mathcal{R}_{i,j,k}^{p,\rho,\varrho}(\xi,\psi,\varphi) = \frac{\mathcal{N}_{i,p}(\xi)\mathcal{M}_{j,\rho}(\psi)\mathcal{L}_{j,\varrho}(\varphi)w_{i,j,k}}{\sum_{\hat{i}=1}^{n_{bf}}\sum_{\hat{j}=1}^{m_{bf}}\sum_{\hat{k}=1}^{l_{bf}} \mathcal{N}_{\hat{i},p}(\xi)\,\mathcal{M}_{\hat{j},\rho}(\psi)\,\mathcal{L}_{\hat{k},\varrho}(\varphi)w_{\hat{i},\hat{j},\hat{k}}}, \tag{2.10}$$

respectively. The curves, surfaces and solids constructed with NURBS basis functions are defined the same as for B-splines. In particular, the basis functions in $1D$ ($\mathcal{N}_{i,p}$), 2D ($\mathcal{N}_{i,p} \times \mathcal{M}_{j,\rho}$) and 3D ($\mathcal{N}_{i,p} \times \mathcal{M}_{j,\rho} \times \mathcal{L}_{j,\varrho}$) in Equations 2.5, 2.6 and 2.7 are replaced by $\mathcal{R}_i^p$, $\mathcal{R}_{i,j}^{p,\rho}$ and $\mathcal{R}_{i,j,k}^{p,\rho,\varrho}$, respectively.

2.3. IGA discretization

From the perspective of numerical analysis, the B-spline (NURBS) entities correspond to meshes where the non-empty knot span $[\xi_i, \xi_{i+1}]^d$ are analogous to the mesh elements, and the control points associated with the basis functions define the shape of the domain. When considering the isoparametric principle, the field in question (e.g., displacement, temperature) is represented using the same basis functions as the geometry, and the coefficients of the basis functions (control variables) correspond to the Degrees of Freedom (DoF).

The construction of the matrix systems proceeds as for traditional FEA. Figure 2.5 illustrates a 2D B-spline entity used to analyze a scalar problem. In this example, the B-splines functions are of second order ($p = 2$) and the mesh has ten elements per spatial dimension. For additional information on using B-spline families for numerical analysis, we refer to [103, 43].

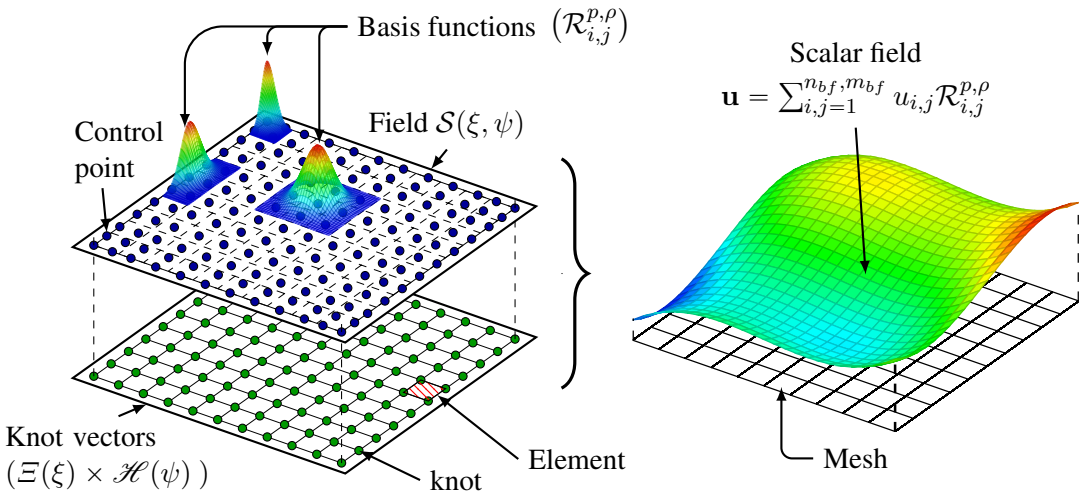


Figure 2.5.: Illustration of a 2D B-spline entity used to analyze a scalar problem (e.g., temperature) throught a Galerkin approximation. In here, $u_{i.j}$ are the DoF.

3. Numerical Solvers

In this chapter, we describe the direct and the iterative solvers we employ to study the impact of continuity reduction on the computational cost. First, we describe the multifrontal direct solver, which is the state-of-the-art method for the direct solution of linear algebraic systems. Second, we introduce the Krylov (sub)space iterative solvers. These are one of the ten numerical methods most influential on science and engineering in the 20th century [52, 56]. In order to better expose the impact of the continuity reduction on the iterative solvers, we focus only on the preconditioned Conjugate Gradients (CG) iterative solver, which is briefly described at the end of the section. Nevertheless, much of the analysis performed for CG solver can be easily generalized to the case of other iterative solvers.

3.1. Direct Solver

There exist several types of direct solvers, such as those based on LU and QR factorizations. The idea behind these solvers is to perform a proper decomposition of the original matrix in terms of the multiplication of two auxiliary matrices leading to two linear systems of equations that can be easily solved. The fastest direct solvers are based on LU factorization (Cholesky factorization in the case of symmetric positive definite systems). These solvers decompose the matrix into a lower-triangular matrix (L) and an upper-triangular one (U) and solve the respective triangular systems sequentially in order to obtain the solution of the original problem.

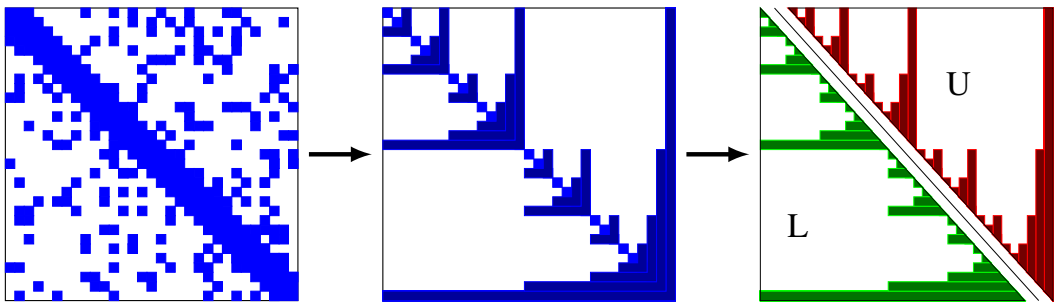


Figure 3.1.: LU decomposition of a sparse matrix.

For sparse systems (which arise in Finite Element Analysis (FEA) due to the local support of the basis functions), a reordering of rows and columns of the matrix is performed before proceeding with the factorization. This minimizes the subsequent fill-in in the L and U factors, thus improving the performance of the solver (Figure 3.1). The matrix is commonly reordered

according to a nested dissection technique [69], since this ordering algorithm is optimal for minimizing the fill-in of the L and U factors for the case of structured grids with an equal number of elements in each spatial direction, and it is quasi-optimal for many other cases.

3.1.1. Multifrontal factorization method

The state-of-the-art implementation is based on the multifrontal solver, a method of solution of sparse linear systems proposed in [53]. This solver is a generalization of the frontal direct solver presented in [81]. The multifrontal solver performs a recursive partitioning of the mesh into pairs of disconnected pieces (subdomains) that are interconnected by small subsets of degrees of freedom called separators (Figure 3.2). Figure 3.3 illustrates an example where a 2D mesh is recursively partitioned four times.

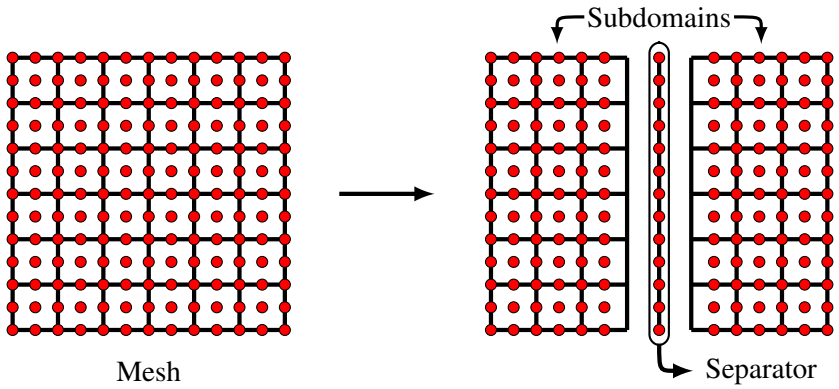


Figure 3.2.: Subdomains resulting from the first recursive partition of a 2D problem. For simplicity, we sketch a finite element discretization using a polynomial order $p = 2$ and C^0 continuity. Red circles represent the nodal degrees of freedom in the system, while black lines denote the mesh skeleton (indicating lower continuity).

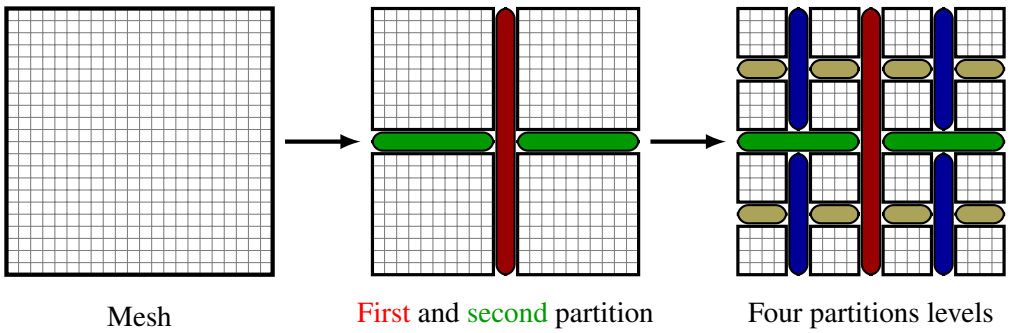


Figure 3.3.: First four recursive partitions of a 2D mesh.

In the factorization process, the elimination of the Degrees of Freedom (DoF) follows the recursive structure of the mesh partition. The unknowns associated to the subdomains are elim-

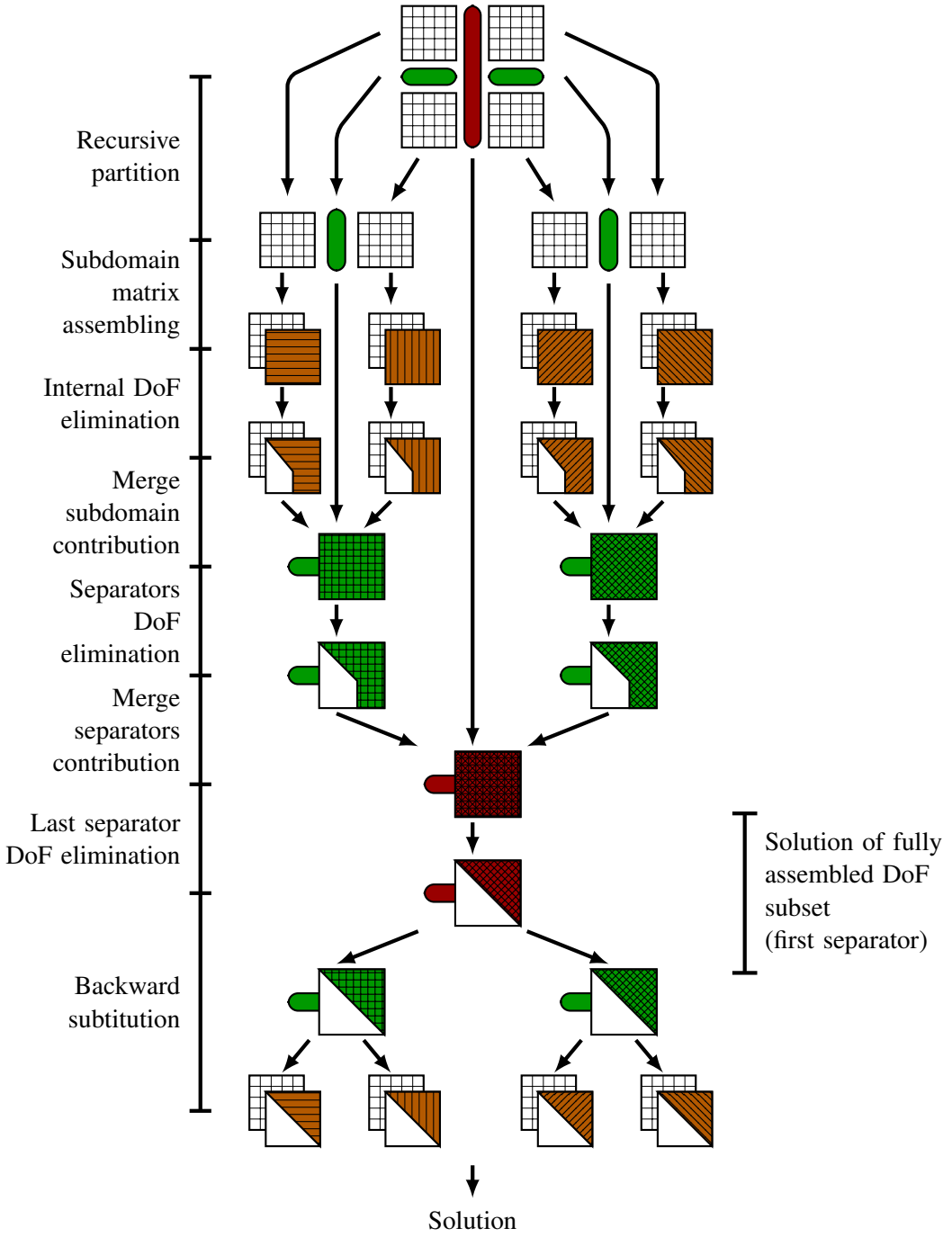
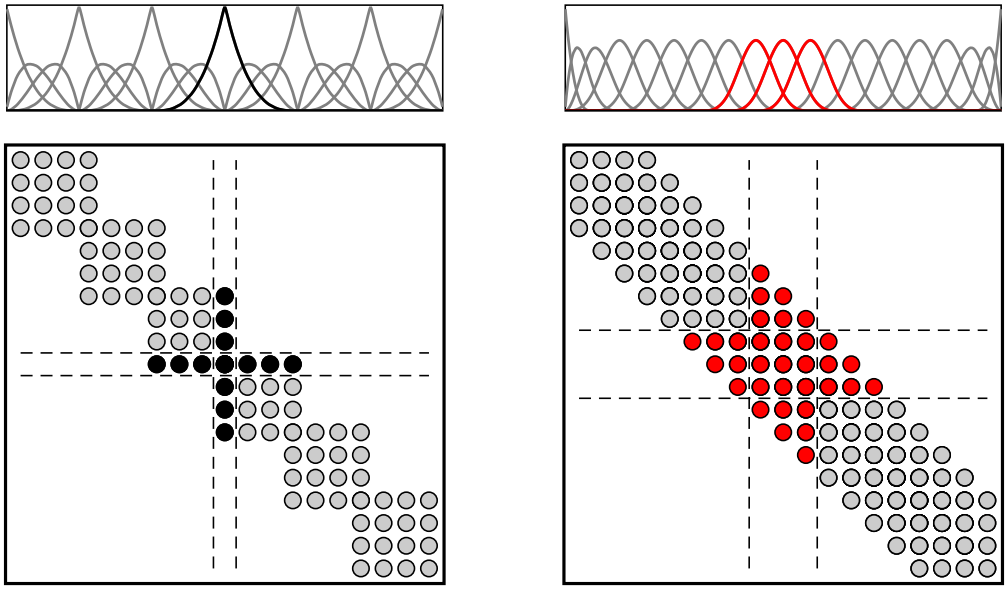


Figure 3.4.: Factorization procedure for a 2D system recursively partitioned into four subdomains.

inated first. Then, the remaining unknowns are eliminated at every separator once they can be fully expressed in terms of the separator DoF. Once all separators are processed, a single subset of fully assembled degrees of freedom (first separator) is solved, and a backward substitution is executed following the structure of the system partition in order to recover the eliminated DoF and solve the problem. An illustration of the factorization procedure and solution of a 2D problem recursively partitioned into two subdomains is presented in Figure 3.4. For additional details on the factorization process, we refer to [81, 53, 33, 100].

The selected discretization heavily influences the performance of the recursive elimination of the system. For traditional FEA, the interconnection between subdomains is weak due to the minimal inter-element continuity. Subdomains are connected by narrow separators (Figure 3.2). In Isogeometric Analysis (IGA), the high inter-element continuity strengthens the interconnection between subdomains, since the growth of the basis function support increments the number of DoF shared between elements. Therefore, wider separators are required to interconnect the subdomains. For instance, in a C^{p-1} IGA system, the separators are p times wider (Figure 3.5).



(a) Traditional FEA (C^0): weak connection

(b) Highly continuous IGA (C^{p-1}): strengthened connection

Figure 3.5.: Illustration of a separator that interconnects two 1D subdomains (using $p = 3$ basis functions). Higher continuous discretizations (C^{p-1} system) involve wider separators and more dof to connect the subdomains. Traditional FEA (C^0 system) benefits from narrow separators.

Hence, highly continuous discretizations degrade the performance of the direct solver per unknown (DoF), increasing time and memory requirements. Indeed, highly continuous IGA is p^3 times more expensive than traditional FEA per unknown, as indicated in Table 1.1 on Chapter 1. Ultimately, the increment in the cost occurs because the size of the separators increases, which

results in more expensive matrix factorizations [41].

3.2. Iterative solver

Nowadays, the most competitive iterative techniques used to solve large-scale linear systems belongs to the Krylov (sub)space methods [108, 52, 56, 75, 73, 93]. Some of the iterative solvers included in this class are the Conjugate Gradient (CG) method presented by Hestenes and Stiefel in [76], the Generalized Minimal RESiduals (GMRES) method developed by Saad and Schultz [109] and the Biconjugate Gradients (BiCG) technique first presented in [86] by Lanczos, which can be interpreted as a variant of the CG [62] for nonsymmetric systems.

The area of Krylov (sub)space methods is in continuous development, and many advances have been performed during the last decades [113]. In general, the advances in this field focus on developing methods to solve particular situations. For instance, the block Krylov iterative methods to solve problems with multiple Right Hand Side (RHS) [74], the Jacobian-free Newton-Krylov (JNFK) methods to deal with non-linear problems and the Newton-Krylov-Schwarz (NKS) methods developed to better approximate compressible potential flows [31] and chemically reacting flows [92]. Moreover, some variants of traditional Krylov methods have been developed to improve particular aspects of those iterative approaches, for example, Biconjugate Gradients stabilized (BiCGSTAB) method was designed to improve the convergence stability of BiCG resulting in a faster method with smoother convergence.

An essential element that makes Krylov (sub)space methods competitive is the use of preconditioners. The idea of preconditioning consists of replacing the original system with an equivalent one that is more suitable for approximating a problem solution with the chosen Krylov space method. In particular, the condition number of the matrix of the preconditioning system is smaller than the original matrix one. This usually results in a speed-up of the convergence of the iterative solvers (reducing the number of iterations).

In practice, the better the preconditioner is speeding up the solver convergence, the more expensive its construction and application becomes. Then, a good preconditioner technique is the one that better balance performance and cost in order to reduce the total solution time. For instance, in a problem where the preconditioner is not reusable, a technique with an economical preconditioner construction cost is more appropriate, considering that the iterative solver must rebuild the preconditioner in each iteration.

There are several types of preconditioners, starting with the matrix splitting techniques used in classic iterative solvers, e.g., Jacobi and successive overrelaxation (SOR) [108]. Additional preconditioners types include the approximate inverse techniques [24, 36], and the multilevel techniques [8, 35]. Most of these preconditioning methods are universally applicable and even if they are not optimal for all kind of problems, they deliver an efficiency improvement of the iterative solvers' performance. A preconditioner rather simple and inexpensive to implement consists on an Incomplete LU (ILU) factorization [108, 23]. This preconditioner belongs to the splitting techniques and has several variants due to its success in speeding up the convergence of many iterative solvers. For additional details on preconditioning techniques, we refer to [23, 35].

In the remaining part of this section, we briefly describe the CG method preconditioned with the ILU technique, which is the approach we will use to expose the impact of continuity reduc-

tion on the iterative solvers. We focus on a single iterative solver considering the complexity required to analyze the impact of the continuity reduction in all the iterative solvers simultaneously. Nevertheless, the analysis performed here can be easily generalized to the case of other preconditioned iterative solvers.

3.2.1. Preconditioned Conjugate Gradient method

The preconditioned Conjugate Gradient (CG) iterative solver is one of the best alternatives to solve boundary value problems that involve symmetric positive definite systems of the form $A\mathbf{x} = \mathbf{b}$ [108, 75]. Here A is the system matrix, and $\mathbf{b}$ is a given RHS vector. The algorithm for this method is presented below

Algorithm 1: Preconditioned Conjugate Gradient (CG)

Input: $\mathbf{b}$, $\mathbf{x}_0$, A, P^{-1}
Compute $\mathbf{r}_0 \coloneqq \mathbf{b} - A\mathbf{x}_0$, $\mathbf{z}_0 \coloneqq P^{-1}\mathbf{r}_0$, **and** $\mathbf{d}_0 \coloneqq \mathbf{z}_0$
for $j = 0, 1, \cdots,$ ***until convergence*** **do**
  $\alpha_j \coloneqq (\mathbf{r}_j, \mathbf{z}_j)/(A\mathbf{d}_j, \mathbf{d}_j)$ `// Step length`
  $\mathbf{x}_{j+1} \coloneqq \mathbf{x}_j + \alpha_j \mathbf{d}_j$ `// Approximate solution`
  $\mathbf{r}_{j+1} \coloneqq \mathbf{r}_j - \alpha_j A\mathbf{d}_j$ `// Residual`
  $\mathbf{z}_{j+1} \coloneqq P^{-1}\mathbf{r}_{j+1}$ `// Preconditioning`
  $\beta_j \coloneqq (\mathbf{r}_{j+1}, \mathbf{z}_{j+1})/(\mathbf{r}_j, \mathbf{z}_j)$ `// Gram-Schmidt constant`
  $\mathbf{d}_{j+1} \coloneqq \mathbf{z}_{j+1} + \beta_j \mathbf{d}_j$. `// Search direction`

In each iteration, the CG method computes a residual $\mathbf{r}_{j+1}$ and a search direction $\mathbf{d}_{j+1}$. The algorithm uses constant α_j to make the new residual orthogonal with respect to the residuals and the search directions in previous steps. Besides, by using the Gram-Schmidt constant β_j, the new search direction becomes A-orthogonal with respect to all previous residuals and search directions. The approach continues iterating until satisfying the convergence criteria which usually is when the Frobenius norm of the residual is smaller than the desired tolerance tol ($||r_{j+1}|| \leqslant tol$). Then, $\mathbf{x_{j+1}}$ can be considered a sufficiently accurate approximation of the problem solution.

The number of sparse matrix-vector products on Krylov spaces methods depends of the number of NonZero (NZ) entries in matrix A. In general, the discretization approach influences the sparsity of system matrix A. A highly continuous IGA discretization delivers a matrix with more NZ entries (denser matrix) than a FEA one with the same number of unknowns. Therefore, when using highly continuous discretizations, the iterative solver becomes more expensive. Indeed, the matrix-vector multiplication is at most eight times more expensive when using a highly continuous IGA discretization instead of a FEA one. This cost becomes $\mathcal{O}(p^2)$ more expensive when comparing the highly continuous IGA discretization with a FEA one that includes static condensation of the internal element DoF [40].

3.2.2. Incomplete LU preconditioning technique

We consider a preconditioner based on an ILU factorization with zero fill-ins. This approach performs a truncated Gaussian elimination that generates a preconditioner matrix P with the same NZ pattern as that of the system matrix A. This technique is based on the IKJ version of Gaussian elimination [108]. The ILU algorithm (algorithm 2) performs the elimination only on the NZ entries, when $a_{ij} \in \mathcal{P}$, being a_{ij} the matrix entry at the i-th row and j-th column and $\mathcal{P}$ the NZ pattern of matrix A.

Algorithm 2: IKJ version of Gaussian elimination

Input: A, N
for $i = 2, \cdots, N$ **do**
  for $k = 1, \cdots, i-1$ **do**
    if $a_{ik} \in \mathcal{P}$ **then**
      $a_{ik} = a_{ik}/a_{kk}$
    for $j = k+1, \cdots, N$ **do**
      if $a_{ij} \in \mathcal{P}$ **then**
        $a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$

The ILU algorithm travels across the matrix by rows (first loop), and at each row, it performs a series of operations (green and blue commands on the algorithm) to construct the LU factors. Figure 3.6 illustrates the procedure of Gaussian elimination conducted on a single row.

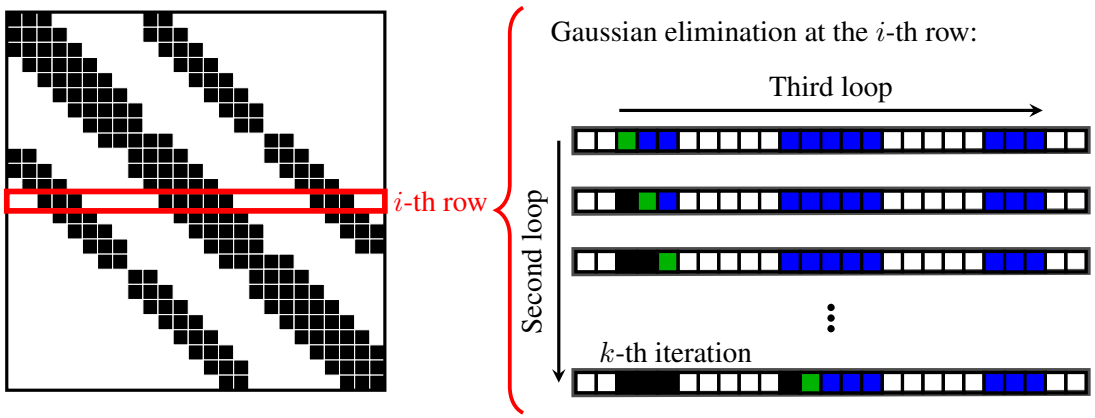


Figure 3.6.: Illustration of the procedure of Gaussian elimination for a single row of the original matrix A. ■ refers to the operations over the matrix diagonal ($a_{ik} = a_{ik}/a_{kk}$), and ■ are the operations over the entries to the left of the diagonal ($a_{ij} = a_{ij} - a_{ik} \cdot a_{kj}$).

The discretization continuity affects the performance of the ILU preconditioner technique. The cost of the preconditioner construction is up to 48 times more expensive when using a highly continuous IGA discretization rather than a FEA one. Moreover, the cost of preconditioning application is up to eight times more expensive [40].

4. refined Isogeometric Analysis

In this section, we propose a discretization strategy that we call refined Isogeometric Analysis (rIGA). This approach seeks the Galerkin discretization that delivers the fastest solution time for a given mesh with a fixed polynomial order. First, we describe the implementation of this method with direct solvers, in particular, with the state-of-the-art multifrontal solver [67, 65]. Second, we detail its variation employed with iterative solvers. In this case, we focus on the Conjugate Gradients (CG) iterative solver preconditioned with the Incomplete LU (ILU) technique [66].

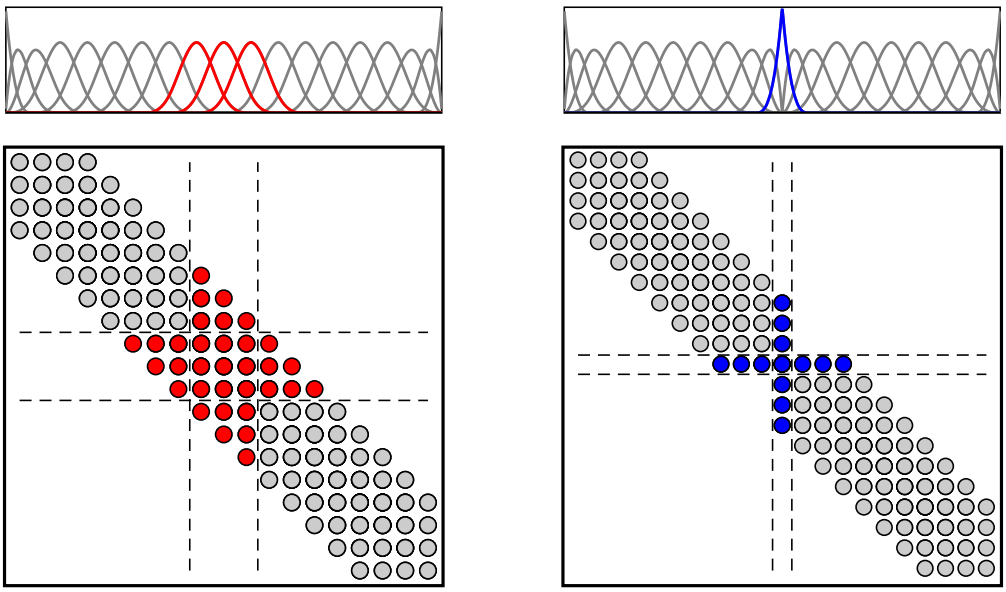
4.1. refined Isogeometric Analysis for direct solvers

The rIGA method intends to decrease the overall cost to solve problems governed by Partial Differential Equations (PDEs) through reducing the continuity degree along the inter-element boundaries while controlling the total number of Degrees of Freedom (DoF) added to the system. More precisely, the method reduces the continuity along the boundaries of the subdomains that result from the recursive partitioning of the mesh performed by the ordering algorithm on the direct solver. The continuity reduction is performed in such a way that it becomes zero in between the subdomains, so the interconnection is weakened (Figure 4.1). Then, rIGA method can be interpreted as a variation of traditional Finite Element Analysis (FEA) that uses C^{p-1} subdomains as elements (macro-elements).

The reduction of continuity narrows the separators that interconnect the subdomains. For instance, in a 2D system with polynomial degree $p = 3$, the separators becomes three times thinner when reducing the continuity degree until zero (Figure 4.2). Thus, the cost of the separators DoF elimination (partial matrix factorization) decreases. Moreover, the reduction of continuity also increases the total number of DoF in the system. The resulting growth in DoF could cause an increment in the total LU factorization cost. Due to this, the reduction of continuity needs to be closely monitored.

To control the overhead paid due to the higher number of DoF, we perform localized reductions of continuity. The optimal continuity reduction decreases the total cost of performing the LU factorization while keeping the total number of DoF under control. In the first instance, we focus on searching for the optimal discretizations (in terms of minimizing the number of FLoating Point Operations (FLOPs) needed to perform the factorization) by using C^{p-1} Isogeometric Analysis (IGA) discretizations enriched by an arbitrary number of C^0-separators.

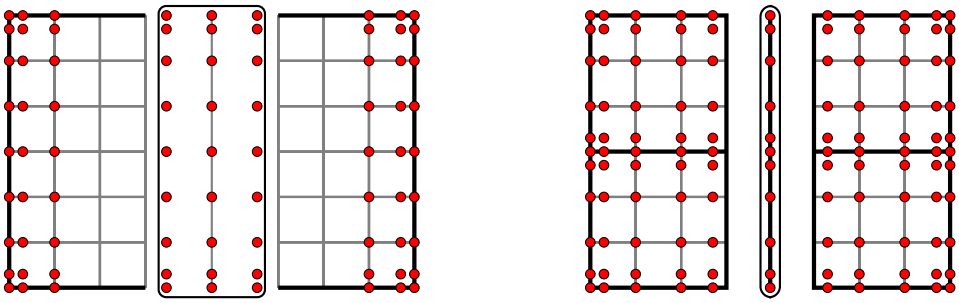
Since the tensor product structure used in Non-uniform Rational Basis Splines (NURBS)-based IGA discretizations limits the continuity reduction over local mesh zones, we adopt a simple implementation that consists of reducing the continuity over hyperplanes that cross the entire mesh in a single direction. The hyperplanes correspond to certain subdomains boundaries (separators). This rIGA implementation starts by building a C^{p-1} IGA discretization (Figure 4.3a).



(a) Highly continuous IGA (C^{p-1}): strengthened connection

(b) rIGA ($C^{p-1} + C^0|_{\text{separator}}$): weaker connection

Figure 4.1.: Illustration of a separator that interconnects two 1D subdomains (using $p = 3$ basis functions). rIGA discretization involves narrower separators than IGA but it increases the number of the total system dof.



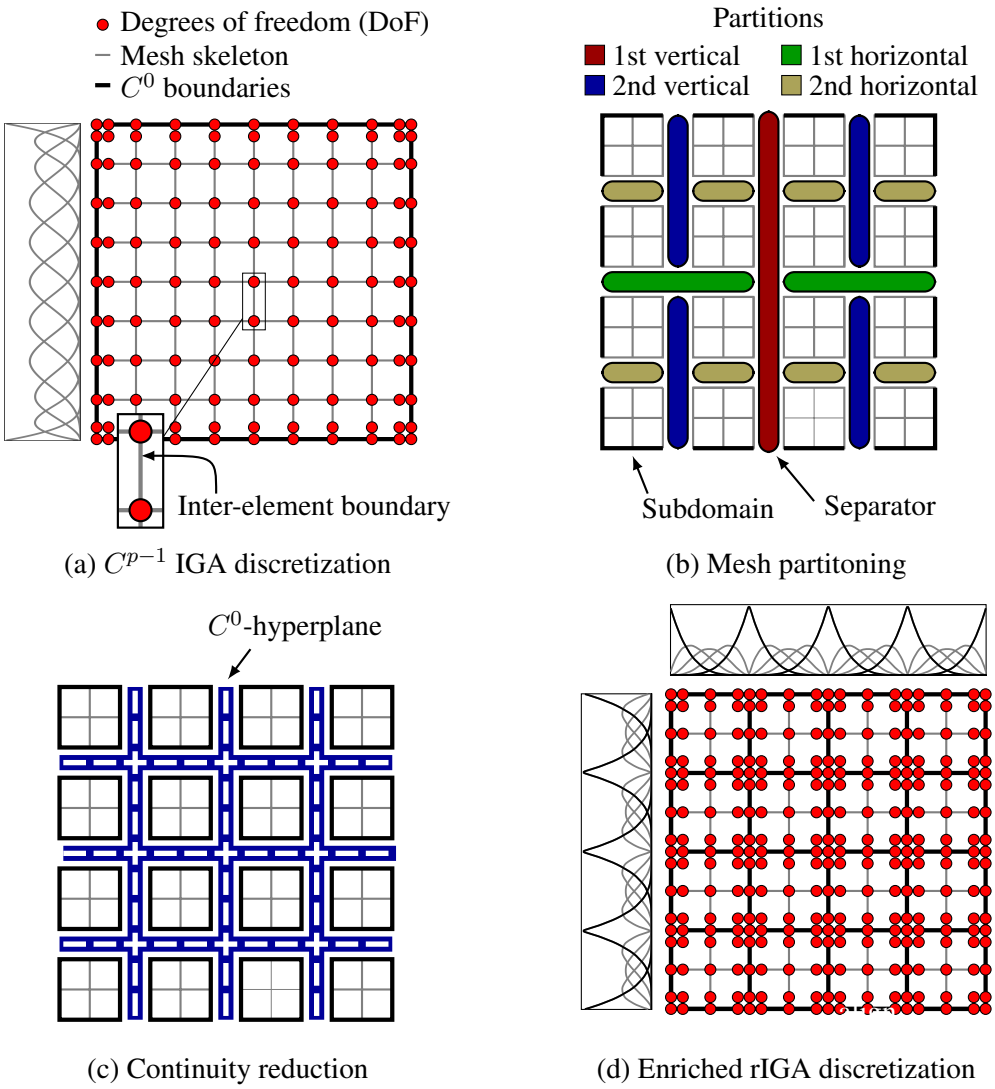
(a) Separator used to partition a C^{p-1} IGA system

(b) Separator used to partition an rIGA system

Figure 4.2.: Illustration of the width reduction and the increment in the number of unknowns of a separator used to interconnect two subdomains that result from partitioning a 2D system with 6×6 elements and polynomial order $p = 3$. In this example, rIGA introduces two C^0-separators.

Next, the ordering algorithm partitions the mesh into submeshes (macro-elements) interconnected by separators (Figure 4.3b). Lastly, we reduce the continuity to zero across the interface

between the subdomains (Figure 4.3c), obtaining an enriched rIGA discretization (Figure 4.3d).



(a) C^{p-1} IGA discretization

(b) Mesh partitoning

(c) Continuity reduction

(d) Enriched rIGA discretization

Figure 4.3.: Illustration of the rIGA implementation structure for a 2D system. In here, red circles represent the nodal degrees of freedom in the system, while gray lines denote the mesh skeleton. Bold black lines represent lower continuity.

4.1.1. Computational complexity for direct solvers

The computational complexity is a measure of the number of FLOPs required to solve a particular problem. In here, we provide theoretical estimates of the computational complexity for a rIGA system solved with direct solvers. This measure explains why rIGA can be several times faster than both IGA and FEA when applied to a fixed mesh size with p greater than one. To

simplify the derivation of the theoretical estimates, we assume that the discretization has the same number of elements in each spatial dimension. Moreover, we consider that the system is solved via a multifrontal direct solver using a nested-dissection technique as ordering method. We briefly introduce the computational complexity for the cases of FEA and IGA since the cost estimates for rIGA are based on those. Then, we provide the cost estimate for rIGA.

4.1.1.1. Cost estimates for finite element and isogeometric analysis

The cost to solve the system of linear equations resulting from discretizing elliptic problems has been previously analyzed in [41, 38, 33]. Those studies used a multifrontal direct solver to compute the solution of the linear systems. This cost consists of three parts. The first part considers the matrix reordering cost, while the remaining two parts are the costs to perform the matrix factorization and backward substitution. We derive the theoretical estimates assuming that the matrix reordering and backward substitution costs are negligible, which is always the case for moderate to large size systems of equations.

In the multifrontal direct solver, the matrix decomposition procedure consists of the partial elimination of the subsets of DoF, either subdomains or separators. The cost to perform a partial LU (Cholesky) factorization of a dense matrix is $\mathcal{O}(q^3)$, with q being the size of the eliminated DoF. The total cost of the matrix decomposition is obtained by adding the contribution of all partial factorizations (Figure 4.4).

The size of the separators (number of unknowns) is

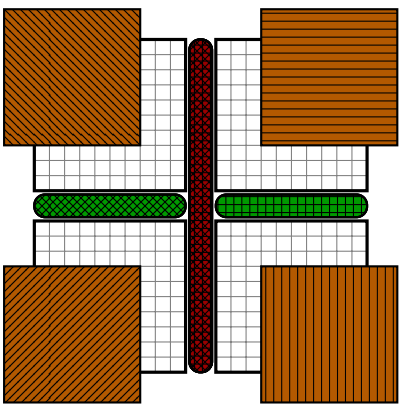
$$q_{sep} = \mathcal{O}\left(N^{(d-1)/d}(k+1)\right),$$

where N is the total number of DoF, d is the dimension, and k is the inter-elemental continuity. We denote the separator continuity k. Thus, $(k+1)$ stands for the separator width, which is 1 in traditional FEA and p for IGA. The size of the minimal subdomains (leaves of the tree, see [100]) is equal to the number of bubble basis functions that every subdomain contains. For traditional FEA, the subsystem size is $q_{sub} = (p-1)^d$, since each subdomain corresponds to one element. In highly continuous IGA, the minimal subdomain size is $q_{sub} \approx 1$ since for collections of $p+1$ elements in each dimension, only one basis function can be eliminated. See [41] for a detailed description of the process.

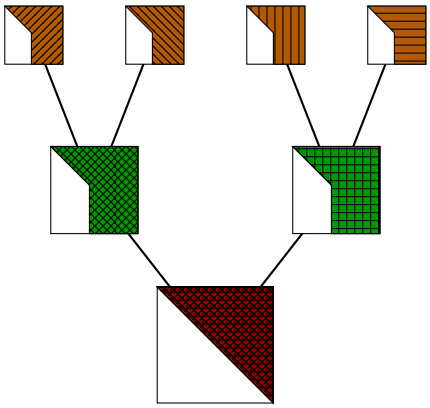
The cost of matrix factorization is then given by

$$\begin{aligned}
\text{Cost} &= \sum_{\eta_{sep}} \mathcal{O}\left((q_{sep})^3\right) + \eta_{sub} \cdot \mathcal{O}\left((q_{sub})^3\right) \\
&= \sum_{\eta_{sep}} \mathcal{O}\left(\left(N^{(d-1)/d}(k+1)\right)^3\right) + \eta_{sub} \cdot \mathcal{O}\left((q_{sub})^3\right) \\
&= \underbrace{\mathcal{O}\left(\left(N^{(d-1)/d}(k+1)\right)^3\right)}_{\text{Skeleton}} + \underbrace{\eta_{sub} \cdot \mathcal{O}\left((q_{sub})^3\right)}_{\text{Interior dof}}, \quad \text{[FLOPs]}
\end{aligned} \tag{4.1}$$

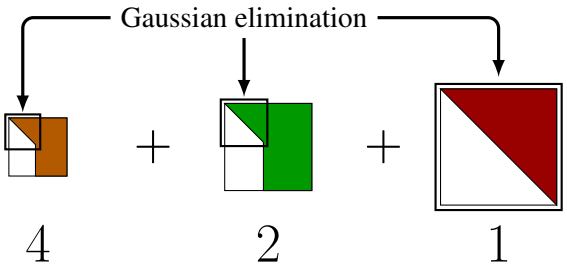
where η_{sep} is the number of separators used to partition the system, and η_{sub} is the number of subdomains. The first term corresponds to the skeleton cost, that is, the cost to eliminate



(a) Subsets of unknowns of a 2D system



(b) Partial matrix decomposition order



(c) Factorization cost

Figure 4.4.: Illustration of the elimination of unknowns of a 2D system partitioned into four subdomains. The cost of factorization is the sum of all partial decompositions $4\mathcal{O}(q^3) + 2\mathcal{O}(q^3) + \mathcal{O}(q^3)$.

the DoF of the separators while the last term is the cost of static condensation, i.e., the cost of eliminating the interior subdomain DoF.

The cost estimates for traditional FEA and highly continuous IGA become:

FEA:

$$\begin{aligned} &\mathcal{O}\left(\left(N^{(d-1)/d}\right)^3\right) + N_{\text{elem}}(p-1)^{3d} \\ &\qquad \approx \mathcal{O}\left(N^{3(d-1)/d}\right) + \mathcal{O}\left(Np^{2d}\right), \end{aligned} \qquad \text{[FLOPs]} \tag{4.2}$$

IGA:

$$\begin{aligned} &\mathcal{O}\left(\left(N^{(d-1)/d}p\right)^3\right) + \eta_{sub}\mathcal{O}(1) \\ &\qquad \approx \mathcal{O}\left(N^{3(d-1)/d}p^3\right), \end{aligned} \qquad \text{[FLOPs]} \tag{4.3}$$

where N_{elem} is the number of elements, and the number of DoF in FEA is $N = \mathcal{O}(N_{\text{elem}}\, p^d)$. These cost estimates match with those derived in [38] and presented here in Table 1.1. Defining $n_{\text{elem}} = N_{\text{elem}}^{1/d}$ as the number of elements in each spatial direction, then, the number of DoF is given by

$$\text{FEA:} \quad N = (n_{\text{elem}}\, p + 1)^d\,,$$
$$\text{IGA:} \quad N = (n_{\text{elem}} + p)^d\,,$$

and the cost estimates to solve a problem with a mesh with a fixed number of elements and a given polynomial order are

$$\text{FEA:} \quad \mathcal{O}\left((n_{\text{elem}}\, p + 1)^{3(d-1)}\right) + \mathcal{O}\left(n_{\text{elem}}^d\, p^{3d}\right), \quad \text{[FLOPs]} \tag{4.4}$$

$$\text{IGA:} \quad \mathcal{O}\left((n_{\text{elem}} + p)^{3(d-1)}\, p^3\right). \quad \text{[FLOPs]} \tag{4.5}$$

In 2D, the solution cost of both C^0 FEA and C^{p-1} IGA is similar to each other (up to lower order terms), while in 3D, C^0 FEA is $\mathcal{O}(p^3)$ more expensive than C^{p-1} IGA. Indeed, the elimination of the skeleton degrees of freedom in 3D requires a large number of FLOPs, specifically, $\mathcal{O}(p^3)$ times the number of FLOPs required to solve the corresponding C^{p-1} IGA system.

4.1.1.2. Cost estimate for rIGA

To compute the cost to solve a rIGA system, we assume that the factorization is performed in two steps (Figure 4.5). First, we eliminate the degrees of freedom contained in the C^{p-1} subsystems (macro-elements). Then, we eliminate the remaining degrees of freedom associated to the C^0-separators.

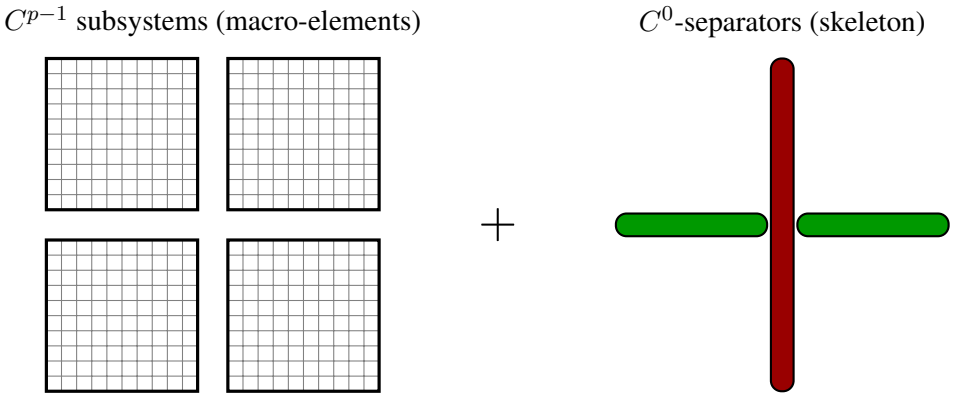


Figure 4.5.: Illustration of C^{p-1} subsystems (macro-elements) and C^0-separators (skeleton) that result from partitioning a 2D system.

The total cost to factorize an rIGA system is given by Equation 4.1 and can be expressed as

$$\underbrace{\eta_{sub} \cdot \mathcal{O}\Big((q_{sub})^3 \Big)}_{C^{p-1} \text{ macro-elements}} + \underbrace{\mathcal{O}\Big(\big(N^{(d-1)/d}(k+1)\big)^3 \Big)}_{C^0 \text{ skeleton}}, \tag{4.6}$$

where the coefficient k in the skeleton term is zero due to the reduction of continuity. At the ℓ-th partition level, rIGA splits the original system into $\eta_{sub} = 2^{d\ell}$ macro-elements of size $N/2^{d\ell}$. The cost of factorizing each of those macro-element is the same as for a C^{p-1} IGA system with size $q_{sub} = (N/2^{d\ell})^{(d-1)/d}\, p$. Then, the total cost becomes

$$\underbrace{2^{d\ell} \cdot \mathcal{O}\left(\left(\frac{N}{2^{d\ell}}\right)^{3(d-1)/d} p^3 \right)}_{C^{p-1} \text{ macro-elements}} + \underbrace{\mathcal{O}\left(N^{3(d-1)/d} \right)}_{C^0 \text{ skeleton}}. \tag{4.7}$$

The continuity reduction enriches the resulting rIGA system. In particular, every cut that splits the original system adds $(p-1)$ DoF in the direction perpendicular to the cut. For instance, in Figure 4.5, the vertical cut (that corresponds to the vertical C^0-separator) adds $(p-1)$ DoF in the horizontal direction, increasing the system size to $(n_{\text{elem}} + p + (p-1))(n_{\text{elem}} + p)$. The horizontal cut adds $(p-1)$ new unknowns in the vertical direction, which increases the total number of DoF in the system to $(n_{\text{elem}} + p + (p-1))^2$. Thus, the number of DoF for a given mesh ($n_{\text{elem}} = 2^g : g \in \mathbb{N}^+$) partitioned into $2^{d\ell}$ macro-elements is

$$\begin{aligned} N = n^d &= (n_{\text{elem}} + p + \underbrace{(2^\ell - 1)(p-1)}_{\text{Enrichment}})^d \\ &= (2^g + p + (2^\ell - 1)(p-1))^d, \end{aligned} \tag{4.8}$$

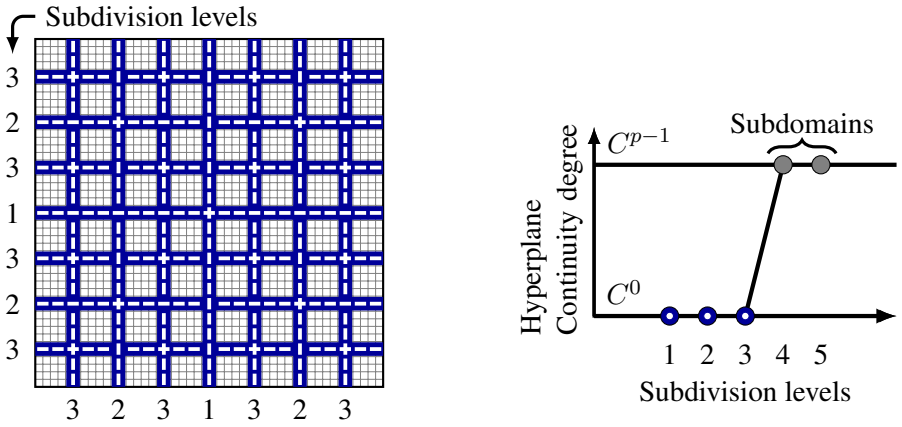
where $(2^\ell - 1)$ is the number of cuts performed in each spatial dimension, and the total cost to factorize an rIGA system is given by

$$\begin{aligned} \theta_{\text{rIGA}} &= \theta_{\text{macro-element}} + \theta_{C^0\text{-separatos}} \\ &= 2^{d\ell} \cdot \mathcal{O}\left(\left(\frac{n}{2^\ell}\right)^{3(d-1)} p^3 \right) + \mathcal{O}\left(n^{3(d-1)} \right) \\ &= \Bigg(\underbrace{2^{(3-2d)\ell} \cdot \mathcal{O}\left(n^{3(d-1)} p^3 \right)}_{\substack{C^{p-1} \text{ macro-elements} \\ \text{contribution}}} + \underbrace{\mathcal{O}\left(n^{3(d-1)} \right)}_{\substack{C^0\text{-separators} \\ \text{contribution}}} \Bigg). \qquad \text{[FLOPs]} \end{aligned} \tag{4.9}$$

4.2. Optimally refined Isogeometric Analysis for direct solvers

In the previous subsection, we proposed a rIGA method that starts with a highly continuous discretization of the classical IGA, and subsequently, when a mesh is being dissected, it introduces hyperplanes that reduce the continuity at the minimum possible degree (C^0) across

certain inter-element boundaries. This approach delivers quasi-optimal discretizations in terms of minimizing the number of FLOPs since it considers only two continuity degrees, C^{p-1} into the macro-elements and C^0 across subdomains interfaces, as illustrated in Figure 4.6.



(a) Hyperplanes and subdomains

(b) Continuity histogram

Figure 4.6.: Ilustration of the continuity degree on a 2D rIGA discretization.

The Optimally refined Isogeometric Analysis (OrIGA) is a version of rIGA that explores all possible continuity degrees on each dissection level (Figure 4.7). While conceptually it would be possible to seek also for the optimal positions to perform the mesh partition, OrIGA focuses only on determining the optimal continuities degrees. In particular, the approach introduces hyperplanes that reduce the continuity to arbitrary degrees across the inter-element boundaries in order to find an optimal discretization. Figure 4.8 illustrates an OrIGA discretization for a $1D$ domain with polynomial degree $p = 3$.

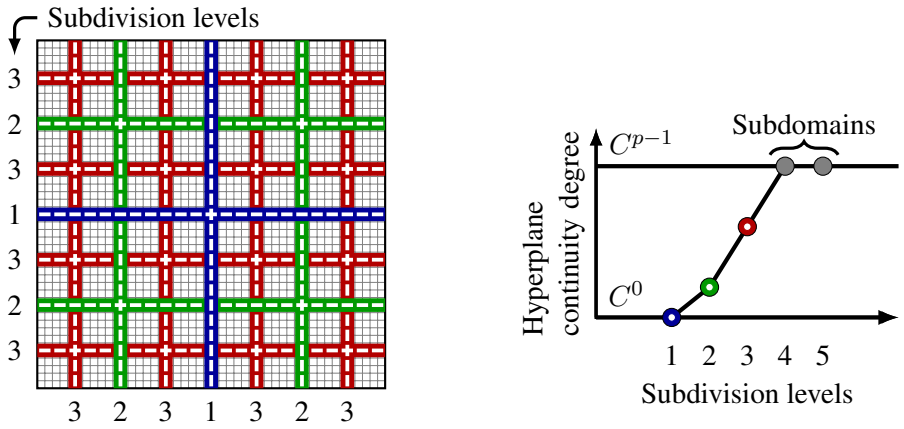
We assume that the mesh is repeatedly bisected using separators across each direction in sequential order. For instance, in 2D, the method partitions the mesh across the vertical direction and then the horizontal direction recursively. After that, we introduce C^k-hyperplanes (that traverse the entire mesh in a single direction) to reduce the continuity until degree k across certain subdomain boundaries. The continuities up to a ℓ-th subdivision level in the horizontal (x) and vertical (y) directions are $\{k_1^x, \ldots, k_\ell^x\}$ and $\{k_1^y, \ldots, k_\ell^y\}$, respectively. Thus,

$$\mathbf{k} = (\mathbf{k}^x, \mathbf{k}^y) = (k_1^x, \ldots, k_s^x, k_1^y, \ldots, k_s^y). \tag{4.10}$$

The OrIGA discretization for a fixed mesh topology and order of approximation p greater than one is given as the solution of the following minimization problem

$$\arg\min_{\mathbf{k}\in\mathcal{S}} \theta(\mathbf{k}), \tag{4.11}$$

where $\theta(\mathbf{k})$ is the computational complexity, and $\mathcal{S}$ is the search space. Following [67, 41], we realize that the cost of the LU factorization is dominated by the cost of eliminating the separators DoF at each partition level. This cost corresponds to the first term in Equation 4.1 that grows in



(a) Hyperplanes and subdomains

(b) Continuity histogram

Figure 4.7.: Ilustration of the continuity degree on a 2D OrIGA discretization.

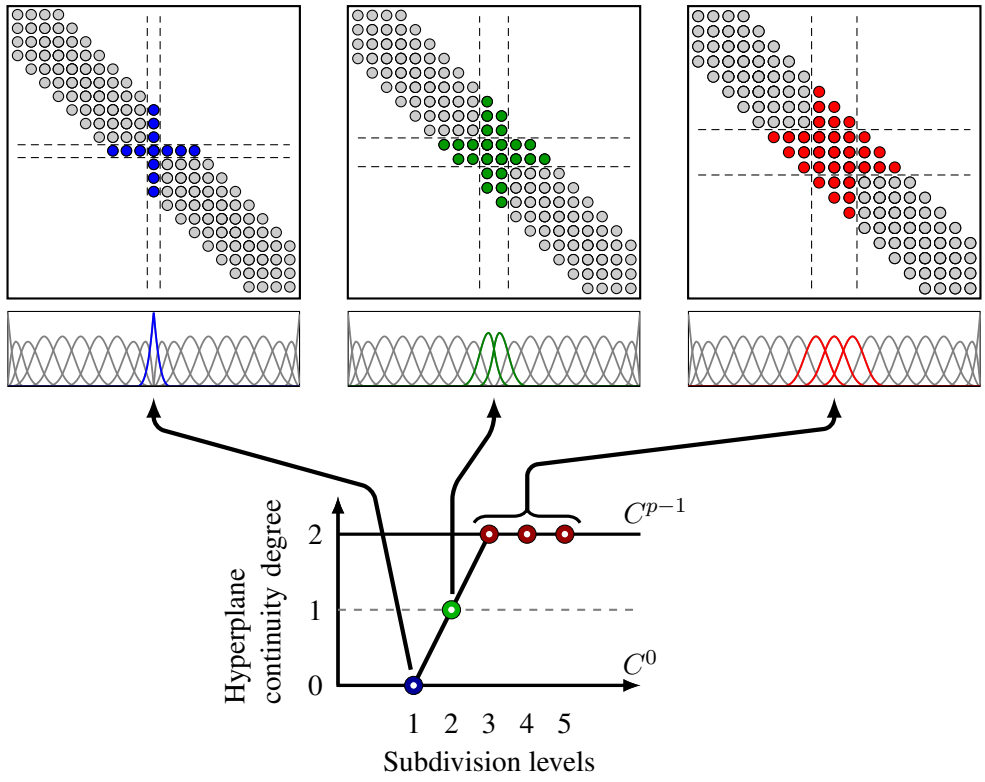
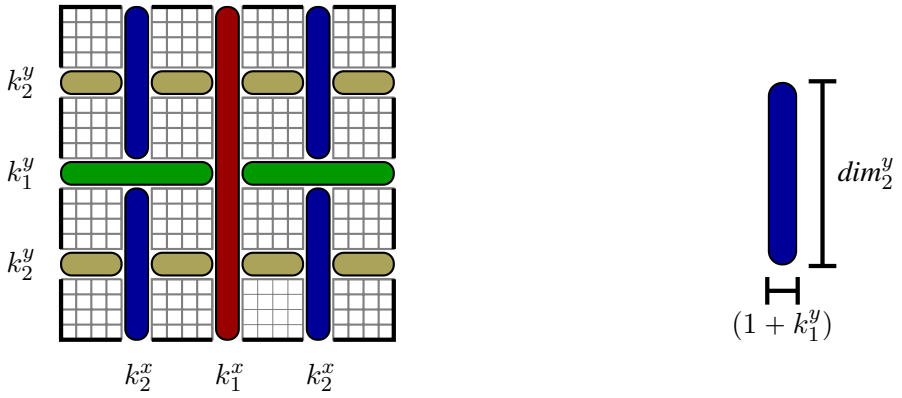


Figure 4.8.: Ilustration of the separators used to interconnects the subdomains on a 1D system using $p = 3$ basis functions.

a cubic fashion with the number of DoF of each separator. For a 2D case, the cost in terms of the partition levels reads as

$$\theta_{sep}(\mathbf{k}) = \sum_{i=1}^{\ell} \eta^y_{sep}|_i (q^y_{sep}(k)|_i)^3 + \eta^x_{sep}|_i (q^x_{sep}(k)|_i)^3 \,, \tag{4.12}$$

where $\eta^y_{sep}|_i$ and $q^y_{sep}(k)|_i$ are respectively the number and size of the separators in the y-direction inserted at the i-th subdivision level. We sum over all the separators in both directions, and the summands in the equation are cubes of dimensions equal to the separators size. Figure 4.9 illustrates the separators size of the second partition level in a 2D case.



(a) Two subdivision levels of a 2D mesh

(b) Separator size $q^y_{sep}|_2$

Figure 4.9.: The number of DoF of one vertical separator in the second subdivision level (blue) is $q^y_{sep}|_2 = dim^y_2(k^x_1 + 1)$. By decreasing the continuity k^x_2, the number of DoF increases in the perpendicular direction, which results in larger (green) separators.

Note that for ℓ subdivision levels, we have $2^\ell - 1$ cuts in every direction. This generates $2^{d\ell}$ submeshes, and the number of separators that split them also grows exponentially. Assuming that the first cut is vertical (associated with an unknown continuity k^x_1), the number of separators at the i-th subdivision level is

$$\begin{aligned} \eta^y_{sep}|_i &= 4^{i-1}, \\ \eta^x_{sep}|_i &= 2 \cdot 4^{i-1}. \end{aligned} \tag{4.13}$$

Moreover, dim^x_i and dim^y_i are equal to

$$\begin{aligned} dim^y_i &= \left(\frac{n_{\text{elem}}}{2^{i-1}} + p^y\right) + \left(\sum_{j=1}^{\ell-i+1} 2^{j-1}(p^y - (k^y_{i+j-1} + 1))\right), \\ dim^x_i &= \left(\frac{n_{\text{elem}}}{2^{i}} + p^x\right) + \underbrace{\left(\sum_{j=1}^{\ell-i} 2^{j-1}(p^x - (k^x_{i+j} + 1))\right)}_{(ii)}, \end{aligned} \tag{4.14}$$

where the second term (ii) in both expressions is the number of DoF that were added to the system by all the continuity reduction on the complementary (perpendicular) cuts that intersect the separators under consideration, see Figure 4.9a. For example, for $\ell = 2$, $i = 1$, the size of dim_1^y associated to the separators at the first vertical cut (red in Figure 4.9a) becomes

$$dim_1^y = n_{\text{elem}} + 4p^y - k_1^x - 2k_2^x - 3, \tag{4.15}$$

since all horizontal separators intersect it.

For general ℓ, by substituting (4.13) and (4.14) into Equation 4.12, we obtain

$$\begin{aligned} \theta_{sep}(\mathbf{k}) &= \sum_{i=1}^{\ell} 4^{i-1} \left(\frac{n_{\text{elem}}}{2^{i-1}} + 2^{\ell-i+1}\, p^y - \sum_{j=1}^{\ell-i+1} 2^{j-1}(k_{i+j-1}^y + 1) \right)^3 (k_i^x + 1)^3 \\ &+ 2 \cdot 4^{i-1} \left(\frac{n_{\text{elem}}}{2^i} + 2^{\ell-i}\, p^x - \sum_{j=1}^{\ell-i} 2^{j-1}(k_{i+j}^x + 1) \right)^3 (k_i^y + 1)^3. \end{aligned} \tag{4.16}$$

We can see that θ_{sep} contains (sixtic) terms of the form $(k_i^x \cdot k_j^y)^3$ with both positive and negative factors. Therefore, the minimizer of θ_{sep} is non-trivial and cannot be in general computed analytically. The objective function (Equation 4.16) is a generalization of (1) in [67], where now the continuities of the separators may be different from zero.

The cost estimate θ_{OrIGA} is completed by adding the cost that comes from static condensation [122, 41]. The static condensation eliminates all DoF interior to the subdomains using Gaussian elimination. The cost of static condensation in 2D ($d = 2$) is given by

$$\theta_{\text{sub}} = 2^{\,d\ell} \left(\left(\frac{n_{\text{elem}}}{2^\ell} + p^x - 1 \right) \left(\frac{n_{\text{elem}}}{2^\ell} + p^y - 1 \right) \right)^3, \tag{4.17}$$

and it does not depend on $\mathbf{k}$. The total cost to factorize an OrIGA system is expressed as

$$\theta_{\text{OrIGA}}(\mathbf{k}) = \theta_{\text{sub}} + \theta_{sep}(\mathbf{k})\,. \qquad [\text{FLOPs}] \tag{4.18}$$

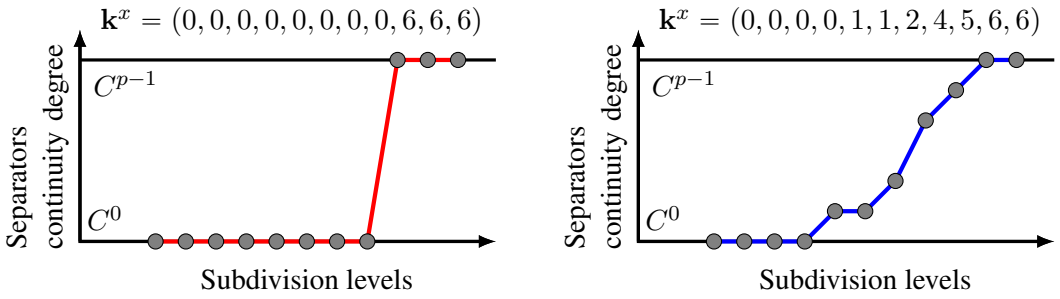
4.2.1. Search space and its reduction

We define the search space $\mathcal{S}$ as a discrete set of possible continuities of the separators in all subdivision levels, that is,

$$\mathcal{S} = \{\mathbf{k}, k_i^z = 0, \ldots, p^z - 1,\ z = x, y,\ i = 1, \ldots, \ell\}, \tag{4.19}$$

for 2D. This space is a generalization of the space of admissible continuities in rIGA, see Figure 4.10.

We aim to minimize θ_{OrIGA} over all possible combinations of continuities $\mathbf{k}$. That is, to find a minimizer that lies in $\mathbb{N}^{d\ell}$, being $d = 2$. This search space is *finite*, unfortunately, the number of combinations grows exponentially with the number of subdivision levels ℓ. More precisely, the number of combinations is $(p^x p^y)^\ell$. Therefore, an exhaustive search is not possible even for a moderate value of ℓ.



(a) rIGA (b) OrIGA

Figure 4.10.: Continuity histograms of the separators used to partition (a) a rIGA mesh and (b) a OrIGA mesh with $\ell = 11$ subdivision levels. The discretizations use a polynomial degree ($p = 7$). While rIGA considers separators of only minimum (0) and maximum $(p-1)$ continuities, OrIGA explores all admissible continuities $\{0, 1, \ldots, p-1\}$ in every subdivision level.

It is necessary to restrict the search space. First, we realize that $k_1^x = 0$ since it appears in Equation 4.16 only once and with positive sign. Additionally, the sequence of optimal continuities in both directions has to be non-decreasing. That is, the vector of continuities $\mathbf{k}$ of Equation 4.10 that minimizes Equation 4.16 satisfies

$$k_i^z \leqslant k_{i+1}^z \quad \textit{for all} \quad z = x, y \quad \textit{and} \quad i = 1, \ldots, s-1. \tag{4.20}$$

To proof this assumption by contradiction, we first consider that $k_i^x > k_{i+1}^x$ for some i. We show that there exists $\hat{\mathbf{k}}$ such that $\theta_{\text{OrIGA}}(\mathbf{k}) > \theta_{\text{OrIGA}}(\hat{\mathbf{k}})$. Define

$$\hat{\mathbf{k}} = (k_1^x, \ldots, k_{i-1}^x, k_{i+1}^x, k_i^x, k_{i+2}^x, \ldots, k_s^x, k_1^y, \ldots, k_s^y) \tag{4.21}$$

There are two kinds of summands in Equation 4.16 that are affected by the switch of k_i^x and k_{i+1}^x. The first type of summand is of the form

$$c_1(c_2 - c_3 k_i^x - 2c_3 k_{i+1}^x), \quad c_1, c_2, c_3 \in \mathbb{N}. \tag{4.22}$$

Note that these numbers depend on i, s, p^x and p^y. However, all these summands decrease when flipping k_i^x with k_{i+1}^x, that is,

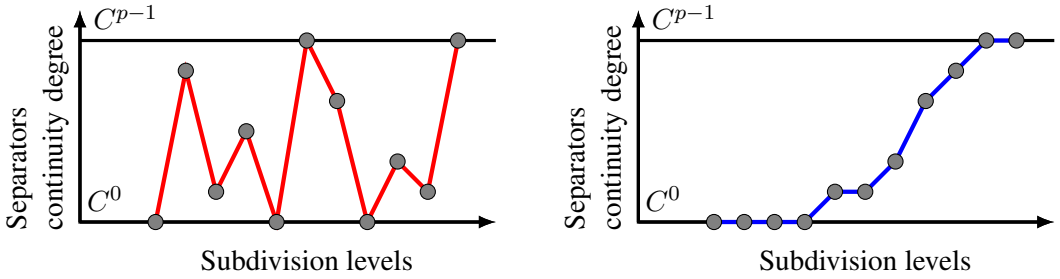
$$k_i^x + 2k_{i+1}^x < k_{i+1}^x + 2k_i^x \tag{4.23}$$

which is equivalent to $k_i^x > k_{i+1}^x$.The other types of summands are the two containing terms $(k_i^x + 1)^3$ and $(k_{i+1}^x + 1)^3$, respectively. The sum of these two summands is

$$4^{i-1}(k_i^x + 1)^3(2L - c_1)^3 + 4^i(k_{i+1}^x + 1)^3 L^3 \quad c_1, L \in \mathbb{N}. \tag{4.24}$$

which again decreases with the change of k_i^x and k_{i+1}^x under the assumption that $k_i^x > k_{i+1}^x$. Therefore $\theta_{\text{OrIGA}}(\mathbf{k}) > \theta_{\text{OrIGA}}(\hat{\mathbf{k}})$, which contradicts that $\mathbf{k}$ is the minimizer and prove that our assumption is appropiate.

By assuming that the continuity sequence in both directions has to be non-decreasing, we introduce a significant reduction of the continuity search space. Observe that while there are p^ℓ possible continuity vectors in one variable, the number of non-decreasing continuity vectors is equal to the number of non-decreasing paths in a rectangular $p \times \ell$ grid, which is only $\binom{p+\ell}{\ell}$, see Figure 4.11. For instance, in a system with $p = 5$ and $\ell = 10$, the cardinality of the reduced 1D space is only 3 003 while in the case of the whole search space it is 9 765 625. For the 2D case, the reduced space size is $3\,003^2$.



(a) $\mathbf{k}$ that cannot minimize (b) Admissible $\mathbf{k}$

Figure 4.11.: Search space reduction. (a) While the space of all possible continuity vectors grows exponentially with the number of subdivision levels ℓ, (b) the reduced search space considers only $\binom{p+\ell}{\ell}$ non-decreasing continuity vectors .

4.2.2. OrIGA implementation

The search for the continuity-aware optimal IGA starts with the rIGA solution. We employ a heuristic approach that uses the following observation from our numerical experiments: rIGA and OrIGA solutions are strongly related. Thus, we use rIGA discretization (represented by the continuity vector $\mathbf{k}^{\text{rIGA}}$) to initialize OrIGA and explore exhaustively only a certain neighborhood of $\mathbf{k}^{\text{rIGA}}$. Let i be the number of subdivision levels where rIGA is enriched by C^0-continuous separators (the "jump" of the rIGA continuity vector). We define the *r-neighborhood* of $\mathbf{k}^{\text{rIGA}}$ as the number of subdivision levels that occurred r subdivisions prior i, and r subdivisions after $i + 1$, see Figure 4.12. In the r-neighborhood, we consider all continuities that satisfy the non-decreasing continuity sequence assumption. Among them, we numerically find the minimizer of Equation 4.16. If not stated differently, we set $r = 2$ in all our experiments.

Remark 1. The computation of the rIGA discretization (continuity vector) comes at a negligible cost. Observe that rIGA considers only C^0 and C^{p-1}-continuous separators that are identical in all directions, and therefore the computation requires only ℓ evaluations of Equation 4.16. We denote by $\mathbf{k}^{\text{rIGA}}$ the rIGA solution.

Figure 4.13 illustrates the continuity vectors of both rIGA and OrIGA discretizations for a system of $N_{\text{elem}} = 1024^2$ elements and polynomial degree $p = 7$. We searched exhaustively the space $\mathcal{S}$ of all feasible continuity vectors, which requires $\binom{7+10}{10}^2 = 19448^2$ evaluations of Equation 4.16. While the exhaustive search of $\mathcal{S}$ required 507 seconds on a laptop equipped with

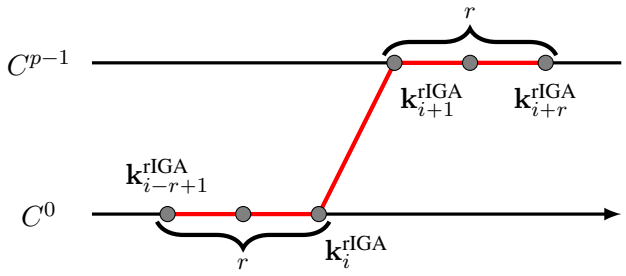


Figure 4.12.: The r-neighborhood of the rIGA solution (red) is shown for $r = 3$. The $2r$ affected continuities of the separators at levels $i - r + 1, \ldots, i + r$ are being optimized to minimize Equation 4.16. The optimal solution must be non-decreasing.

a 2.20GHz processor, the computation of rIGA discretization took few milliseconds (2.6^{-4}) and the search of its 2-neighborhood only 0.75 seconds.

The OrIGA solution (continuity vector) differs from rIGA only by a few coordinates. This phenomenon applies to various degrees and mesh sizes as we show in Chapter 5. For the analyzed cases, the OrIGA and rIGA continuity vectors differ at most at two coordinates. Moreover, this difference appears in the neighborhood of the continuity "jump" of the rIGA continuity vector. Therefore, we use the solution obtained by rIGA to initialize the refined exhaustive search.

The search for the optimal continuity discretization is summarized in Algorithm 3. Regarding the approximation quality, highly continuous IGA discrete spaces are strictly contained in both the rIGA and OrIGA spaces, so the best approximation error of OrIGA is smaller or equal than that of IGA.

Algorithm 3: Optimally refined Isogeometric Analysis OrIGA implementation

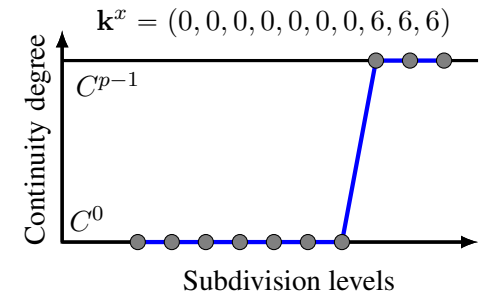
Objective: find $\mathbf{k} = (\mathbf{k}^x, \mathbf{k}^y)$ that minimizes Equation 4.16, i.e., $\arg\min_{\mathbf{k} \in S} \theta(\mathbf{k})$

Input: number of elements n_{elem}, polynomial degree p, search neighborhood r,

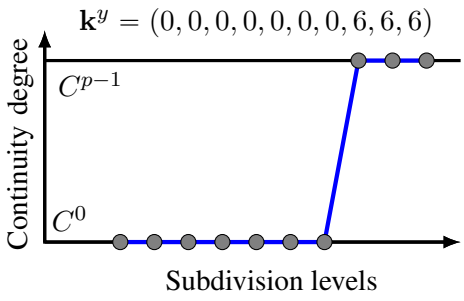
---

Initialize $\mathbf{k}$ by rIGA solution $\mathbf{k}^{\text{rIGA}}$
$F_{min} := F(\mathbf{k}^{\text{rIGA}})$
for $i = 1, \cdots, (\binom{p+2r}{p})$ **do**
  for $j = 1, \cdots, (\binom{p+2r}{p})$ **do**
    build non-decreasing $\mathbf{k}_i^x$ and $\mathbf{k}_j^y$
    if $F_{min} > F(\mathbf{k}_i^x, \mathbf{k}_j^y)$ **then**
      $\mathbf{k} := (\mathbf{k}_i^x, \mathbf{k}_j^y)$
    else
      $F_{min} := F(\mathbf{k}_i^x, \mathbf{k}_j^y)$

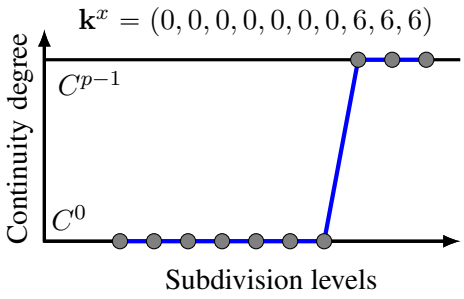
Output: OrIGA continuity vector $\mathbf{k}$.



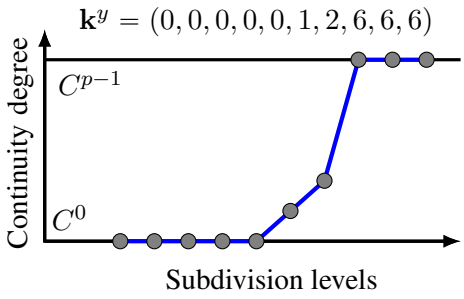
(a) rIGA: Horizontal separators



(b) rIGA: Vertical separators



(c) OrIGA: Horizontal separators



(d) OrIGA: Vertical separators

Figure 4.13.: Continuity vectors resulting from the global exhaustive search of the optimal continuity space for a 2D mesh consisting of 1024^2 elements, polynomial degree $p = 7$ and $\ell = 10$ subdivision levels. The continuity vector of OrIGA in the (horizontal) x-direction, $\mathbf{k}^x$, is identical to the rIGA solution, while $\mathbf{k}^y$ differs from rIGA by only two coordinates.

4.3. Refined Isogeometric Analysis for iterative solvers

In here, we propose the variation of the rIGA to use with iterative solvers, in particular, to use with a CG method preconditioned with the ILU factorization technique that produces zero fill-ins. The discretization method starts by partitioning the mesh of C^{p-1} discretizations into macro-elements using an arbitrary number of C^0-hyperplanes. These hyperplanes involve a reduction of continuity in such a way that it weakens the interconnection between the subdomains (macro-elements), as illustrated before in Figure 4.1. Moreover, the use of exclusively C^0-hyperplanes makes the explanation of the method easily tractable. Figure 4.14 illustrates the partitioning of a mesh into four subdomains.

After partitioning the mesh into submeshes, the method performs a static condensation in *all* the macro-elements. This consists in a partial LU (Cholesky) factorization that eliminates the DoF inside every macro-element and results in a reduced system which involves only the degrees

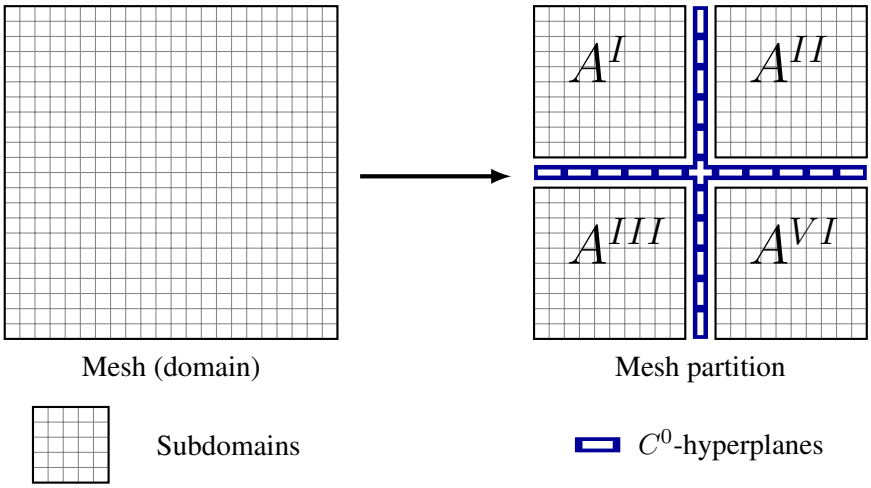


Figure 4.14.: Partition of a 2D mesh into four subdomains (macro-elements).

of freedom at the macro-element boundaries. That is, we compute the Schur complement for *all* the macro-elements and build the skeleton problem by assembling all these Schur complements. Considering the system $A^i\mathbf{x}^i = \mathbf{b}^i$ associated to the degrees of freedom of the i-th macro-element, we perform a partial LU factorization $A^i = L^i\,U^i$, which results in

$$\begin{aligned} \begin{pmatrix} A^i_{int,int} & A^i_{int,bnd} \\ A^i_{bnd,int} & A^i_{bnd,bnd} \end{pmatrix} \begin{pmatrix} \mathbf{x}^i_{int} \\ \mathbf{x}^i_{bnd} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}^i_{int} \\ \mathbf{b}^i_{bnd} \end{pmatrix} \\ \begin{pmatrix} L^i_{int,int} & 0 \\ L^i_{bnd,int} & I \end{pmatrix} \begin{pmatrix} U^i_{int,int} & U^i_{int,bnd} \\ 0 & S^i \end{pmatrix} \begin{pmatrix} \mathbf{x}^i_{int} \\ \mathbf{x}^i_{bnd} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}^i_{int} \\ \mathbf{b}^i_{bnd} \end{pmatrix}, \end{aligned} \tag{4.25}$$

where the subscripts *int* and *bnd* refers to the DoF located in the interior and at the boundaries of the macro-element, respectively. S^i corresponds to the Schur complement of the A^i matrix, which is defined as

$$S^i = A^i_{bnd,bnd} - A^i_{bnd,int}\left(A^i_{int,int}\right)^{-1} A^i_{int,bnd}\,. \tag{4.26}$$

The reduced right-hand side $\mathbf{y}^i_{bnd}$ for the i-th macro-element is computed using Equation 4.27.

$$\begin{pmatrix} L^i_{int,int} & 0 \\ L^i_{bnd,int} & I \end{pmatrix} \begin{pmatrix} \mathbf{y}^i_{int} \\ \mathbf{y}^i_{bnd} \end{pmatrix} = \begin{pmatrix} \mathbf{b}^i_{int} \\ \mathbf{b}^i_{bnd} \end{pmatrix}, \tag{4.27}$$

and the reduced system for the i-th macro-element is

$$S^i\mathbf{x}^i_{bnd} = \mathbf{y}^i_{bnd}\,, \tag{4.28}$$

where $i = 1, ...\eta_{sub}$, being $\eta_{sub} = \eta_{\text{m-e}}$ the number of macro-elements. The reduced systems are assembled all together obtaining the skeleton system $A_{skl}\,\mathbf{x}_{skl} = \mathbf{y}_{skl}$, which corresponds to the degrees of freedom located along the macro-element boundaries (C^0 skeleton mesh). Figure 4.15 illustrates the static condensation step.

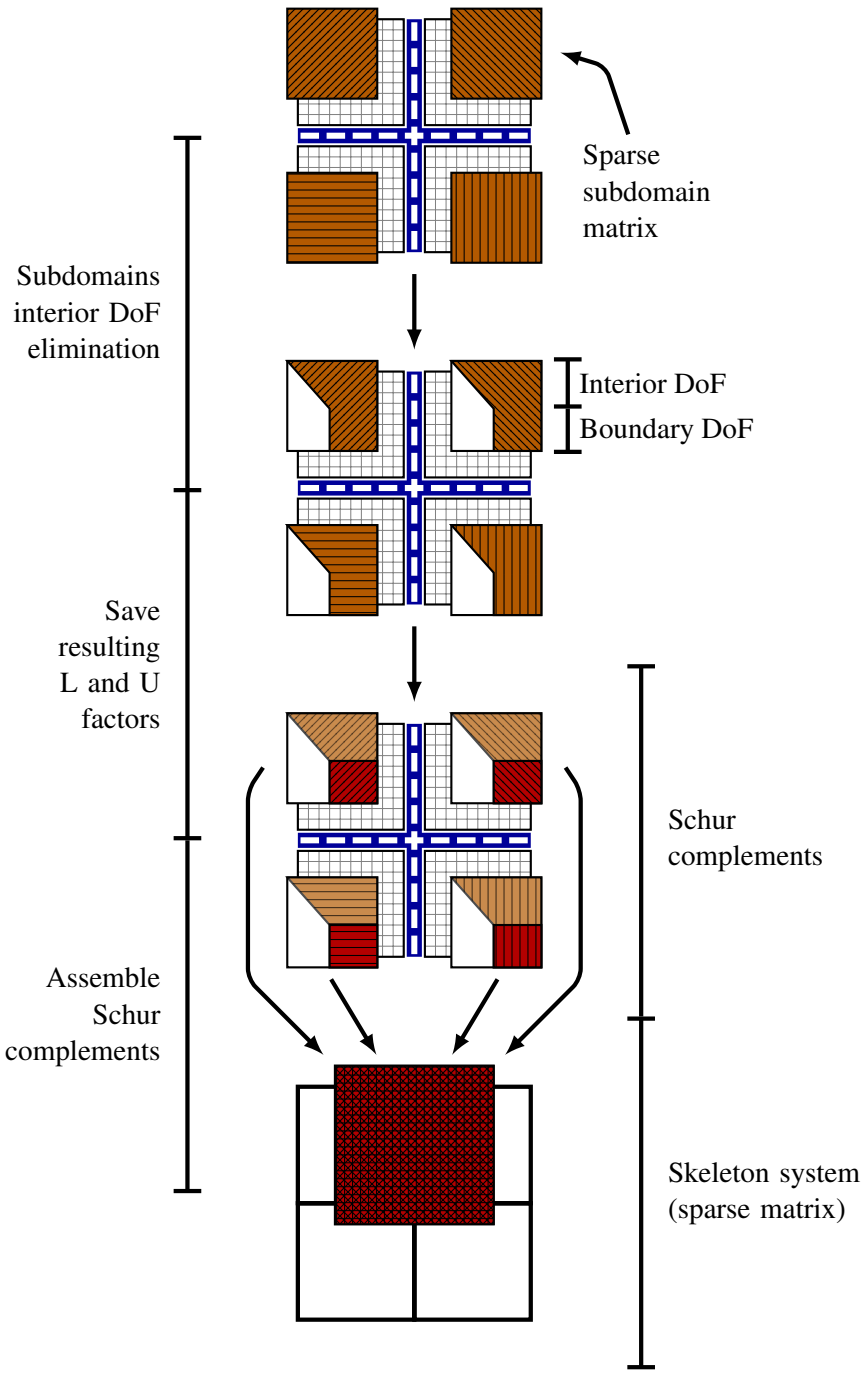


Figure 4.15.: Static condensation procedure for a 2D system recursively partitioned into four subdomains.

We solve the skeleton system using the preconditioned CG iterative solver [76] (algorithm 1). First, we compute the preconditioner matrix P using the ILU factorization technique [108] (algorithm 2). Subsequently, we solve the preconditioned system. Lastly, we perform a backward substitution to obtain the solution of the original system. The backward substitution employs the factors obtained in the static condensation step. In summary, rIGA for iterative solvers can be considered as a hybrid solver strategy that combines a direct solver (static condensation step) to build the Schur complements of the macro-elements, with an iterative method to solve the skeleton system.

4.3.1. Computational complexity for iterative solvers

The computational complexity (total number of FLOPs) of this hybrid solver strategy consists of the sum of the computational costs corresponding to each step, namely, static condensation, preconditioner matrix construction, and skeleton system solution. We separately analyze the computational cost corresponding to each of these steps. Also, to simplify the analysis of the cost, we assume that the mesh is fixed with the same number of elements in each spatial dimension and an order of approximation greater than one ($p > 1$).

4.3.1.1. Cost of static condensation (macro-elements interior DoF elimination)

We partition the mesh into $\eta_{\text{m-e}}$ macro-elements (Figure 4.14). Each macro-element corresponds to a C^{p-1} IGA system of size equal to $N_{\text{elem}} = (s+p)^d$, being d the dimension and s^d the number of elements into the macro-element (Figure 4.16).

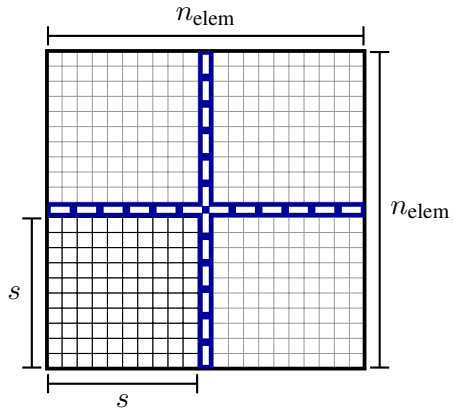
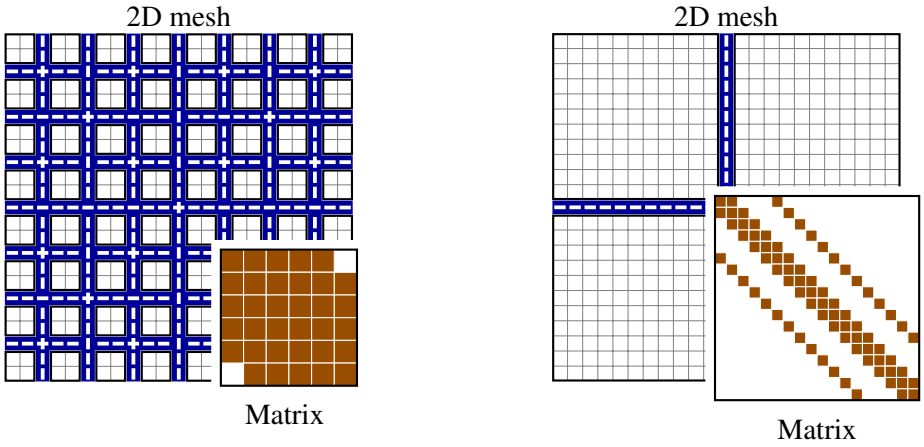


Figure 4.16.: Illustration of a 2D mesh partitioned into four subdomains. The total number of elements is $N_{\text{elem}} = n_{\text{elem}}^d$, while the number of elements in each subdomains is s^d, being $d = 2$ the spatial dimension.

Depending on the size of the macro-element, we use a different approach to estimate the number of FLOPs required to compute the reduced system. For small macro-elements sizes that involve nearly dense matrices, the cost is identical as that of performing the elimination of the

interior degrees of freedom for a dense matrix problem. However, when the macro-elements are large, the corresponding cost is similar as computing the Schur complement for a sparse matrix.



(a) Small macro-element (nearly dense matrices)

(b) Large macro-element (sparse matrices)

Figure 4.17.: Illustration of the static condensation of the interior degrees of freedom (DoF) for a single macro-element.

We realize that the macro-elements containing a number of elements $s^d \leqslant (p+1)^d$, being $(p+1)^d$ the support of the C^{p-1} basis functions, involve matrices nearly dense (Figure 4.17a). In those cases, the cost of static condensation is the same as that of performing a partial factorization on a dense matrix of size N. The cost to perform the partial factorization on the i-th macro-element is given by

$$\begin{aligned} \varphi_{SC}^{i} &= \sum_{k=1}^{N|_{\mathrm{o}}} \left(\sum_{i=k+1}^{N} \left(1 + \sum_{j=k+1}^{N} 2 \right) \right) \\ &= \frac{2}{3} \left(N|_{\mathrm{o}} \right)^3 + \left(\frac{1}{2} - 2N \right) \left(N|_{\mathrm{o}} \right)^2 + \left(2 \left(N \right)^2 - N - \frac{1}{6} \right) N|_{\mathrm{o}}, \end{aligned} \tag{4.29}$$

where $N|_{\mathrm{o}} = (s+p-2)^d$ is the number of interior DoF. This expression counts the number of operations (FLOPs) required to perform Gaussian elimination of the interior macro-elements DoF [68].

Macro-elements of size greater than $s^d > (p+1)^d$ involve sparse matrices (Figure 4.17b). In those cases, the cost of static condensation as that of a multifrontal technique used to compute the Schur complement. That is,

$$\psi_{SC}^{i} \approx \sum_{j=1}^{\ell} \left(\sum_{k=1}^{d} \left(2^{k-1} \cdot 2^{d(j-1)} \overbrace{\left(\underbrace{2^{-(j+k-2)} \left(\sqrt[d]{N|_{\mathrm{o}}} - (p-1) \right)}_{\text{Length}} \right)^{3(d-1)} \underbrace{\left(p^3 \right)}_{\text{Thickness}}}^{\text{Separator size}} \right) \right), \tag{4.30}$$

where ℓ is the number of partition levels. Term $(p-1)$ refers to the number of DoF belonging to the separator that was removed by previous partition levels. For the case of 2D problems, the equation simplifies to

$$\psi^i_{SC}|_{2\mathrm{D}} \approx \sum_{j=1}^{\ell} \left(2^{2(j-1)} \left(2^{-(j-1)} \left(\sqrt{N|_{\mathrm{o}}} - (p-1) \right) \right)^3 p^3 + 2^{2(j-1)+1} \left(2^{-(j)} \left(\sqrt{N|_{\mathrm{o}}} - (p-1) \right) \right)^3 p^3 \right). \tag{4.31}$$

The first separator involves a cost of $(N|_{\mathrm{o}})^{3/2} p^3$ since we did not perform any prior partition. Therefore, the total cost becomes

$$\psi^i_{SC}|_{2\mathrm{D}} = \mathcal{O}\left((N|_{\mathrm{o}})^{3/2} p^3 + \frac{1}{2} \left(3 - 5 \cdot 2^{-\ell} \right) \left(\sqrt{N|_{\mathrm{o}}} + (p-1) \right)^3 p^3 \right). \tag{4.32}$$

Finally, the estimate of number of FLOPs required by the static condensation of the i-th macro-element in 2D is

$$\theta^i_{SC}|_{2D} = \begin{cases} \varphi^i_{SC}|_{2D} & \forall\, s \leqslant p+1 \\ \psi^i_{SC}|_{2D} & \forall\, s > p+1, \end{cases}$$

that is

$$\theta^i_{SC} = \begin{cases} \frac{2}{3} (N|_{\mathrm{o}})^3 + \left(\frac{1}{2} - 2N \right) (N|_{\mathrm{o}})^2 + \left(2 (N)^2 - N - \frac{1}{6} \right) N|_{\mathrm{o}} & \forall\, s \leqslant p+1 \\ \mathcal{O}\left((N|_{\mathrm{o}})^{3/2} p^3 + \frac{1}{2} \left(3 - 5 \cdot 2^{-\ell} \right) \left(\sqrt{N|_{\mathrm{o}}} + (p-1) \right)^3 p^3 \right) & \forall\, s > p+1, \end{cases} \tag{4.33}$$

Ultimately, the cost of the static condensation step consists of the number of FLOPs required to eliminate the interior DoF in *all* macro-elements. This cost is given by

$$\theta_{SC} = \sum_{i=1}^{\eta_{\text{m-e}}} \theta^i_{SC}, \qquad [\text{FLOPs}] \tag{4.34}$$

where $\eta_{\text{m-e}} = (n_{\text{elem}}/s)^d$, being n^d_{elem} the total number of elements of the original matrix. For a fixed macro-element size, the above step scales linearly with respect to the total problem size.

We do not consider the number of operations required to compute the reduced right-hand-side and assemble the skeleton system in θ_{SC}, since their cost is negligible in comparison to the total cost of the hybrid solver strategy.

4.3.1.2. Cost of preconditioning using ILU factorization technique

The number of FLOPs needed to set up the preconditioner depends of the number of NonZero (NZ) entries of matrix A_{skl} in the skeleton system. We realize that the number of NZ entries in A_{skl} corresponds to that of a statically condensed C^0 FEA discretization with a matrix that has the same number of DoF on each macro-element interface. Figure 4.18 illustrates both the full and statically condensed C^0 FEA (right) and the original and statically condensed rIGA (left) discretizations for a 2D problem.

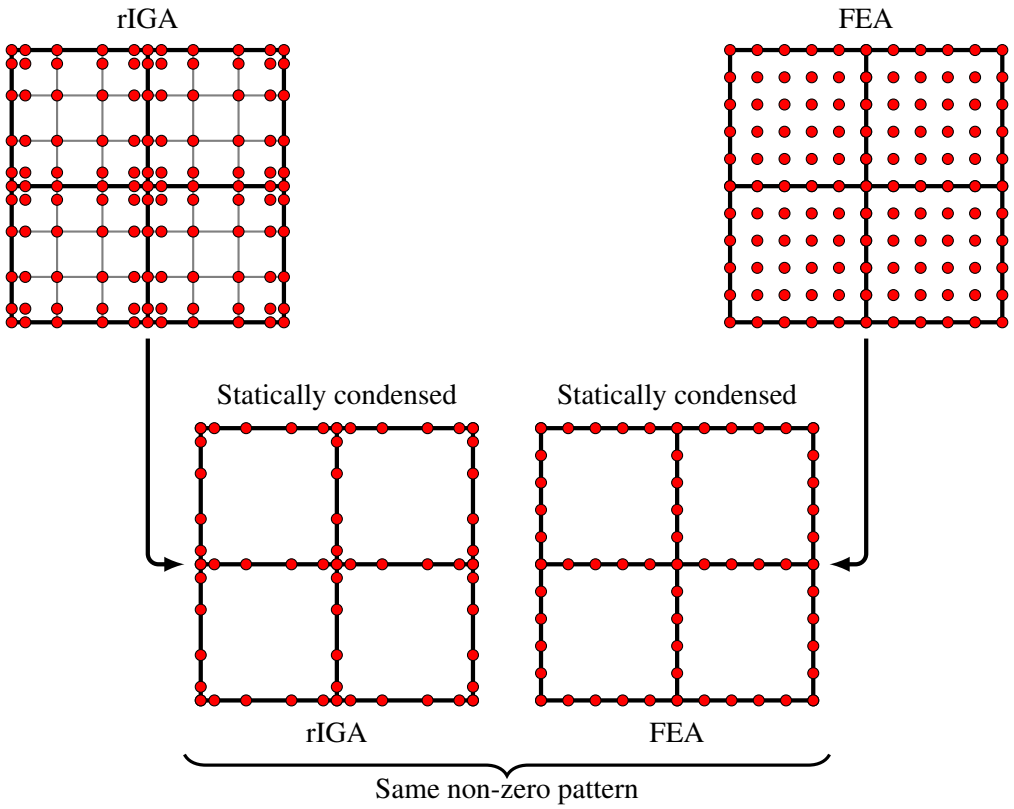


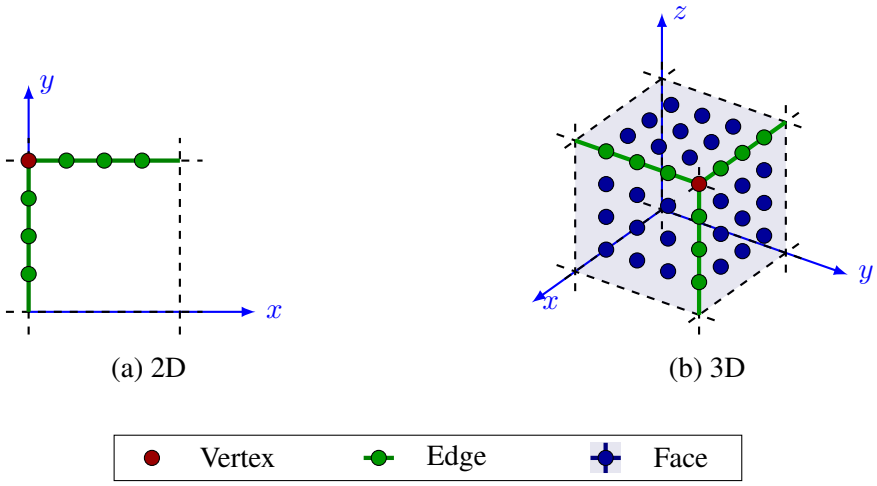
Figure 4.18.: Illustration of both FEA and rIGA discretizations of a 2D system. The rIGA discretization composes of 2×2 subdomains, 6×6 elements and polynomial basis functions of order $p = 3$, while the FEA discretization consists of 2×2 elements and polynomial degree $\hat{p} = 5$. Blue circles represent the nodal degrees of freedom in the system, while black lines denote the mesh skeleton. Bold lines represent C^0 continuity.

The polynomial degree of the C^0 FEA system is $\hat{p}$ and is given by

$$\begin{aligned} \text{FEA element size} &= \text{rIGA subdomain size} \\ (\hat{p}-1)^d &= (s+p-2)^d \\ \hat{p}-1 &= s+p-2 \\ \hat{p} &= s+p-1, \end{aligned}$$

while the number of elements is equal to the number of macro-elements (subdomains) in which the original C^{p-1} system was partitioned. The size of those FEA elements corresponds to the number of interior degrees of freedom in a single macro-element.

The number of NZ entries in the j-th matrix row corresponds to the number of nodes (unknowns) with which the j-th node interacts. Depending on the location of this node, the number of interactions nodes varies [99, 39]. Figure 4.19 illustrates the possible location of the nodes over the elements of a statically condensed C^0 FEA system. Moreover, Table 4.1 presents the number of nodes at each location as well as the number of interactions per node.



(a) 2D (b) 3D

Figure 4.19.: Illustration of the nodes locations over a single element in both 2D and 3D after static condensation.

We consider an ILU with zero fill-ins preconditioner. This approach performs a truncated Gaussian elimination that generates a preconditioner matrix with the same non-zero pattern as that of the original one. We assume that the ILU is based on the *IKJ* version of Gaussian elimination shown in [108] and illustrated in Figure 3.6. This technique proceeds as in algorithm 2.

The number of FLOPs executed when performing a truncated Gaussian elimination in a j-th row is given by

$$\theta_{A_{skl}|_j} = \frac{1}{4}\left(3\, nnz(A_{skl}|_j)^2 - 2\, nnz(A_{skl}|_j) - 1\right), \tag{4.35}$$

where $nnz(A_{skl}|_j)$ refers to the number of NZ entries in the j-th row of matrix A_{skl}. The number of connectivities (interactions) associated to the domain interior nodes are summarized in Table 4.1. We assume that the nodes in the interior of the domain are contained into $(n_{\text{elem}}/s-1)^d$

Dimension	Type	# nodes	# interactions ($nnz\left(A_{skl}|_{\text{row}}\right)$)
2D	Vertex	1	$(2\hat{p}+1)^2 - 4(\hat{p}-1)^2$
	Edge	$2(\hat{p}-1)$	$(2\hat{p}+1)(\hat{p}+1) - 2(\hat{p}-1)^2$
3D	Vertex	1	$(2\hat{p}+1)^3 - 8(\hat{p}-1)^3$
	Edge	$3(\hat{p}-1)$	$(2\hat{p}+1)^2(\hat{p}+1) - 4(\hat{p}-1)^3$
	Face	$3(\hat{p}-1)^2$	$(2\hat{p}+1)(\hat{p}+1)^2 - 2(\hat{p}-1)^3$

Data for a single element.

Table 4.1.: Summary table of the nodes interactions for both 2D and 3D statically condensed C^0 FEA systems [39].

macro-elements and that the remaining nodes only contribute with lower order terms ($L.O.T.$) to the total cost. Then, the number of FLOPs required to build the ILU preconditioner is given by

$$\theta_{pre} = \begin{cases} \left(\frac{n_{\text{elem}}}{s} - 1\right)^d \left(2(\hat{p}-1)\,\theta_{A_{skl}|_{\text{Edge}}} + (1)\,\theta_{A_{skl}|_{\text{Vertex}}}\right) & 2D, \\ \left(\frac{n_{\text{elem}}}{s} - 1\right)^d \left(\begin{array}{r} 3(\hat{p}-1)^2\theta_{A_{skl}|_{\text{Face}}} + 3(\hat{p}-1)\theta_{A_{skl}|_{\text{Edge}}} \\ + (1)\,\theta_{A_{skl}|_{\text{Vertex}}} \end{array}\right) & 3D. \end{cases} \tag{4.36}$$

For a fixed macro-element size, the cost of building the ILU with zero fill-ins preconditioner scales linearly with respect to the total problem size.

4.3.1.3. Cost of the CG iterative solver

The major cost when solving a system with an iterative solver comes from the matrix-vector products. Each of those matrix-vector product consists of two operations, one sum and one multiplication, per NZ entry of the sparse matrix [39]. The total number of operations is proportional to the number of NZ entries in both the system and the preconditioner matrix. Therefore, the cost associated to solve the skeleton system is given by

$$\theta_{it} = \mathcal{O}\left(2\left(nnz\left(P\right) + nnz\left(A_{skl}\right)\right)\right), \tag{4.37}$$

where P is the preconditioner and A_{skl} is the skeleton matrix. Both the skeleton and preconditioner matrix have the same number of NZ entries, which allows to express the cost as

$$\theta_{it} = \mathcal{O}\left(4\,nnz\left(A_{skl}\right)\right), \qquad [\text{FLOPs}] \tag{4.38}$$

The estimate of the number of NZ entries is computed based on Table 4.1, and given by

$$nnz\left(A_{skl}\right) = \begin{cases} \eta_{\text{m-e}}\left(2(\hat{p}-1)\left((2\hat{p}+1)(\hat{p}+1)-2(\hat{p}-1)^2\right)\right. & \\ \qquad \left. +\,(1)\left((2\hat{p}+1)^2-4(\hat{p}-1)^2\right)\right) & 2D \\ & \\ \eta_{\text{m-e}}\left(3(\hat{p}-1)^2\left((2\hat{p}+1)(\hat{p}+1)^2-2(\hat{p}-1)^3\right)\right. & \\ \quad +3(\hat{p}-1)\left((2\hat{p}+1)^2(\hat{p}+1)-4(\hat{p}-1)^3\right) & 3D. \\ \qquad \left. +\,(1)\left((2\hat{p}+1)^3-8(\hat{p}-1)^3\right)\right). & \end{cases} \tag{4.39}$$

As it occured with the static condensation and the preconditioner matrix construction, the cost of matrix-vector multiplication scales linearly with respect to the total problem size for a fixed macro-element size.

5. Numerical results

In this chapter, we present the numerical results for the three types of refined Isogeometric Analysis (rIGA) considered in this Dissertation: rIGA and Optimally refined Isogeometric Analysis (OrIGA) for direct solvers, and the rIGA hybrid solver strategy for iterative solvers. The corresponding implementations use the library PetIGA, which is a high-performance software platform for Isogeometric Analysis (IGA) [48] based on PETSc [11, 10]. PetIGA has been used to model many engineering applications since its inception [48, 119, 116, 41, 38, 33, 39, 119, 116, 26, 42, 55, 25, 111, 110, 54]. For instance, the implementation of nonlinear hyperelastic models for slightly compressible materials with IGA in [25], and the development of a new phase-field model concept with IGA to study polycrystalline solidification as well as related physical phenomena [119].

In the case of rIGA and OrIGA for direct solvers, we use the sequential version of the multifrontal solver MUMPS [4, 5] to solve the matrix system that results from the problem discretization. We select the automatic choice of partitioning technique made by MUMPS, resulting in a METIS [83] algorithm for all the cases.

For the hybrid solver strategy, we perform the static condensation of the internal macroelements Degrees of Freedom (DoF) with the sequential version of the solver MUMPS [4, 5] with METIS [83, 84]. Afterwards, the reduced (skeleton) system resulting from static condensation is solved using the PETSc version of Conjugate Gradients (CG) iterative solver preconditioned with the Incomplete LU (ILU) technique that produces zero fill-ins.

The computational tests of rIGA for direct and iterative solvers are performed in sequential on TACC Stampede system. We use the machine nodes outfitted with turbo boost 2.7 GHz cores (up to peak 3.5 GHz in turbo mode) and 1TB of memory. For the case of OrIGA, all computational tests were solved sequentially on TACC Lonestar5 system. In this case, we select the nodes equipped with 2.3 GHz cores and 512TB of memory. For additional details on TACC machine, we refer to Texas Advanced Computing Center (TACC) home web page (http://www.tacc.utexas.edu).

We perform all computations in sequential in order to report the global FLoating Point Operations (FLOPs) and computational times (in seconds) for several variations in mesh size and polynomial order. In the case of rIGA for direct solvers, we also show the memory requirements (that correspond to the NonZero (NZ) entries in factors L and U expressed in Mbytes) to factorize the matrix system. Moreover, we provide the number of NZ entries of the original matrix A and the skeleton matrix A_{skl} and estimates of the computational cost of the hybrid solver strategy in 3D.

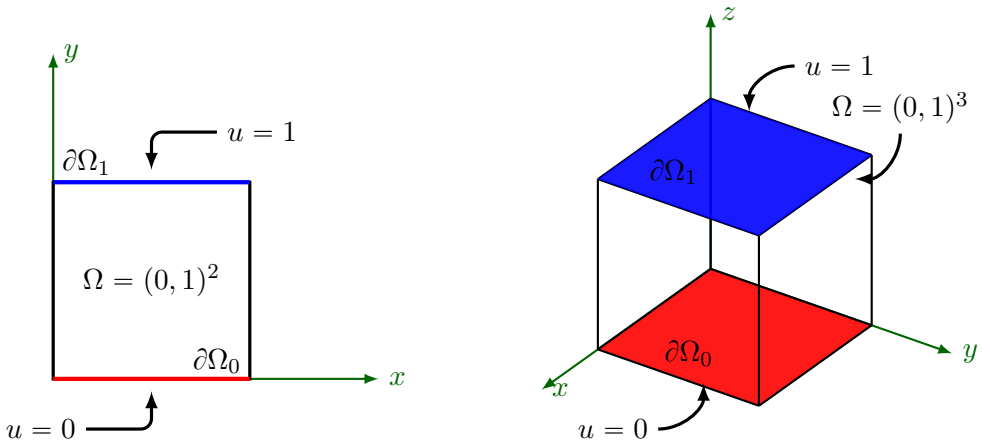
5.1. rIGA for direct solvers

5.1.1. Model problem

We use the Laplace equation in (5.1) as a model problem to exemplify the performance of rIGA for direct solvers. Thus, our problem is given by:

$$\begin{cases} \text{Find } u \text{ such that:} & \\ \nabla \cdot (\nabla u) = 0 & \text{in } \Omega \\ u = 1 & \text{on } \partial\Omega_1 \\ u = 0 & \text{on } \partial\Omega_0 \\ \nabla u \cdot n = 0 & \text{on } \partial\Omega_w, \end{cases} \tag{5.1}$$

where $\Omega = (0,1)^d$, with d being the dimension, $\partial\Omega_w \cup \partial\Omega_0 \cup \partial\Omega_1 = \partial\Omega$, $\partial\Omega_w \cap \partial\Omega_0 = \varnothing$ and $\partial\Omega_w \cap \partial\Omega_1 = \varnothing$. The problem domains for 2D and 3D are illustrated in Figure 5.1.



(a) Domain $\Omega = (0,1)^2$ (b) Domain $\Omega = (0,1)^3$.

Figure 5.1.: Model problem domain.

5.1.2. Implementation details

We use unmapped B-splines to build the model problem considering that the domain is unitary. The tensor product of the unmapped B-splines defines the domain of the problem, obtaining a regular and structured mesh. This resulting meshes incorporate the same number of elements (n_{elem}) in each spatial dimension in order to obtain a uniform structure when partitioning the system. We implement mesh sizes of 512^2, 1024^2 and 2048^2 elements to discretize the model problem in 2D. In 3D, we use three mesh sizes of 32^3, 64^3 and 128^3 elements. Moreover, we use four polynomial degrees p ranging from 2 to 5 to perform the discretization. These polynomial orders are kept constant in each problem.

For every mesh size, we consider a range of cases with particular number of partition levels using C^0-separators. The first case assumes no reduction of continuity, i.e., no level of partition uses C^0-separators, which corresponds to the conventional tensor-product IGA. The last case involves a reduction of the continuity along *all* the inter-elements boundaries, i.e., all levels of partition employ C^0-separators, which correspond to the conventional Finite Element Analysis (FEA). This allows us to analyze the impact of the local reduction of continuity in the computational cost, besides finding the optimal continuity reduction for rIGA.

5.1.3. Fit of estimates

In order to fit the theoretical estimates (Equation 4.9) with the computed number of FLOPs required to factor the systems, we introduce two constants, namely, *A* and *B*, as follows:

$$\theta_{\text{rIGA}} = A\, 2^{(3-2d)\ell} \left(n^{3(d-1)}p^3\right) + B\, \left(n^{3(d-1)}\right)$$

$$2D\text{:}\quad \theta_{\text{rIGA}} = \left(A2^{-\ell}p^3 + B\right)\left(n_{\text{elem}} + p + (2^\ell - 1)(p-1)\right)^3 + L.O.T.\,, \quad \text{[FLOPs]}$$

$$3D\text{:}\quad \theta_{\text{rIGA}} = \left(A2^{-3\ell}p^3 + B\right)\left(n_{\text{elem}} + p + (2^\ell - 1)(p-1)\right)^6 + L.O.T.\,, \quad \text{[FLOPs]}$$

where $L.O.T.$ stands for lower order terms. For each p, we estimate *A* and *B* by solving a least square fitting problem for a large mesh size. Specific values of *A* and *B* are presented in Table 5.1. Constants *A* and *B* are almost independent of the polynomial order, but depend on

Constants		Polynomial order p			
		2	3	4	5
2D	*A*	24.5	21.5	21.5	20.0
	B	26.5	24.5	22.5	20.0
3D	*A*	6.6	6.3	6.0	5.7
	B	7.8	7.2	7.2	7.2

Table 5.1.: Fitting constants computed for every polynomial order in 2D and 3D.

the problem dimension. Both the contribution of forming the Schur complements and the total number of FLOPs performed by LAPACK to factorize the system [27] are included in these constants.

5.1.4. Numerical results

5.1.4.1. FLOPs

We first analyze the FLOPs. Figures 5.2 and 5.4 show the number of FLOPs required to factor the algebraic system for 2D and 3D, respectively. The number of FLOPs is plotted with respect to the macro-element size ($s = n_{\text{elem}}/2^\ell$), being ℓ the number of partition levels that splits the mesh domain into $\eta_{\text{m-e}}$ macro-elements. The size of the macro-elements diminishes as we add C^0-separators. Thus, the C^0 FEA case corresponds to a macro-element size equal to one, while

the C^{p-1} IGA case corresponds to the largest macro-element size. These numerical results confirm the following:

a The theoretical estimates approximate well the numerical results.

b The optimal discretization (in terms of minimizing the number of FLOPs of the direct solver) is in an intermediate stage between uniformly global continuity C^0 and C^{p-1}. Neither of the two extreme cases provides optimal discretizations.

c The reduction factor in the number of FLOPs by using an optimal discretization is approximately $(p+1)^2$ for 2D and p^2 for 3D.

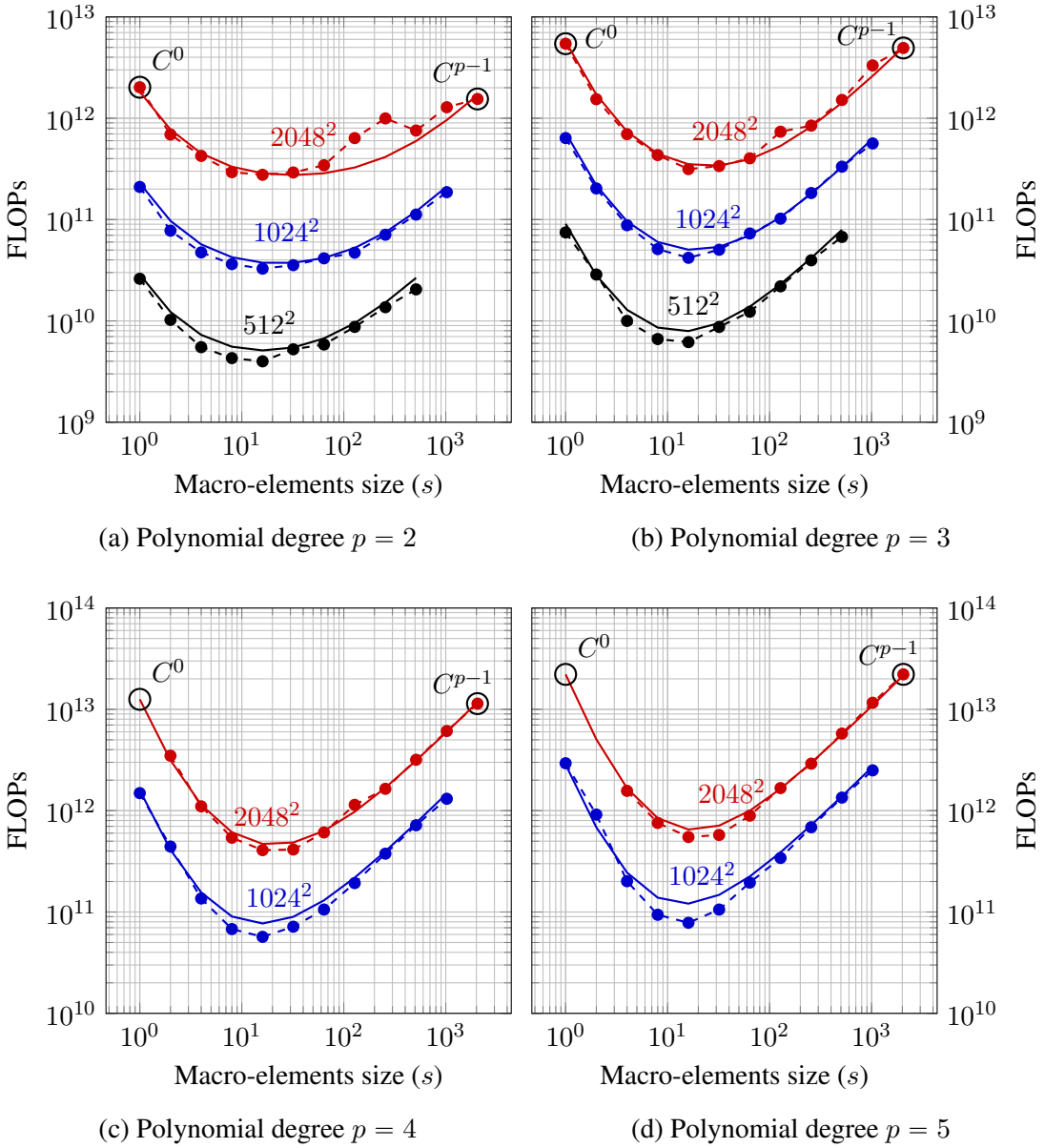
In Figure 5.2, rIGA shows a maximum reduction factor of the number of FLOPs of 40 with respect to C^{p-1} IGA. This reduction factor is obtained when we solve the model problem with $N_{\text{elem}} = 2048^2$ elements and a polynomial order $p = 5$. Further numerical results showed that for a problem with $N_{\text{elem}} = 2048^2$ elements and a polynomial order $p = 9$, rIGA reduces the number of FLOPs by a factor of at least 70 with respect to C^{p-1} IGA, and 83 with respect to C^0 FEA (Figure 5.3) when using solver MUMPS [4, 5].

The discrepancy between the numerical experiments and the theoretical estimates plots occurs since the lower order terms that we exclude from the theoretical estimates are relevant for small mesh sizes (pre-asymptotic regime). However, once we get close to the asymptotic regime, the $L.O.T.$ are negligible. Thus, the estimates fit better the numerical results as observed for the largest mesh sizes in Figure 5.2.

For the problem in Figure 5.3, we should be able to provide results for rIGA cases with deeper partition levels including C^0-separators, e.g., cases with macro-element sizes of $s = 16$ and $s = 8$. However, due to software limitations, it is not possible to compute those cases with PETSc and MUMPS. The standard PETSc installation uses by default 32 bit indices. This configuration stops working once the system matrix of the problem contains more than $2^{31} - 1$ (approximately 1.6e+9) NZ entries on a single process. This limit is independent of the available physical memory of the machine. For the problem with $N_{\text{elem}} = 2048^2$ elements and a polynomial order $p = 9$, the rIGA cases with $s \leqslant 16$ exceed this PETSc limit.

By setting-up PETSc with 64 bit indices, the limit in the number of NZ entries grows. Nevertheless, since MUMPS does not support 64 bit indices, we can not use this package to perform the matrix factorization. PARDISO [102, 101] is a sparse direct solver package that supports 64 bit indices. This package uses a combination of supernode techniques [112] to perform the matrix factorizations. Assuming that the performance of PARDISO is approximately the same as than of MUMPS, we use this package to provide the computational cost for the cases we can not solve using PETSc+MUMPS. In Figure 5.3, the red dashed line with rounded markers corresponds to the cases solved with MUMPS (- ●-) while the dashed line in blue with circle markers refers to the cases solved with PARDISO (- ⊖-). The number of FLOPs required to solve the C^{p-1} IGA case is approximately the same for both MUMPS and PARDISO, being approximately 1.2e+14 FLOPs. We do not compute rIGA cases with $s < 8$ since those cases do not contain any relevant information.

In 3D, the maximum reduction factor of the number of FLOPs observed when using rIGA is about 26 with respect to C^{p-1} IGA, and even larger with respect to C^0 FEA (Figure 5.4). This



(a) Polynomial degree $p = 2$

(b) Polynomial degree $p = 3$

(c) Polynomial degree $p = 4$

(d) Polynomial degree $p = 5$

Figure 5.2.: Number of FLOPs required to eliminate the DoF in the 2D model problem (when using the multifrontal direct solver). The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

reduction factor was analytically obtained because the C^{p-1} IGA case cannot be computed due to memory limitations.

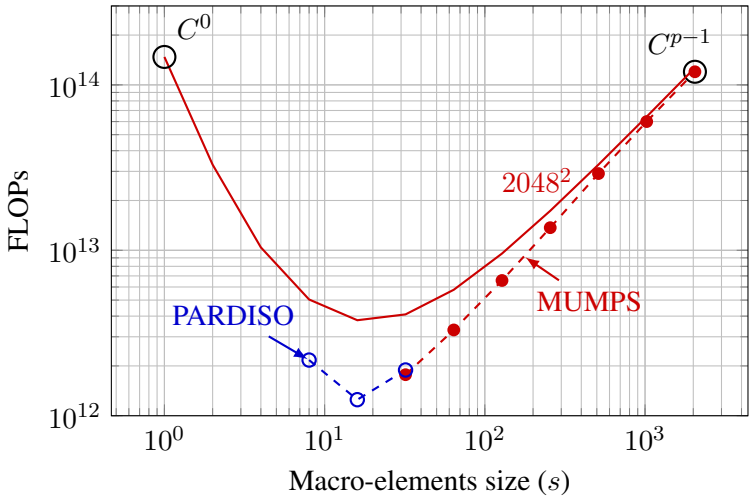


Figure 5.3.: Number of FLOPs required to eliminate the DoF in the 2D model problem discretized with $N_{\text{elem}} = 2048^2$ elements and $p = 9$. The problem is solved with MUMPS (- ●-) and PARDISO (- ⊙-) solvers. The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

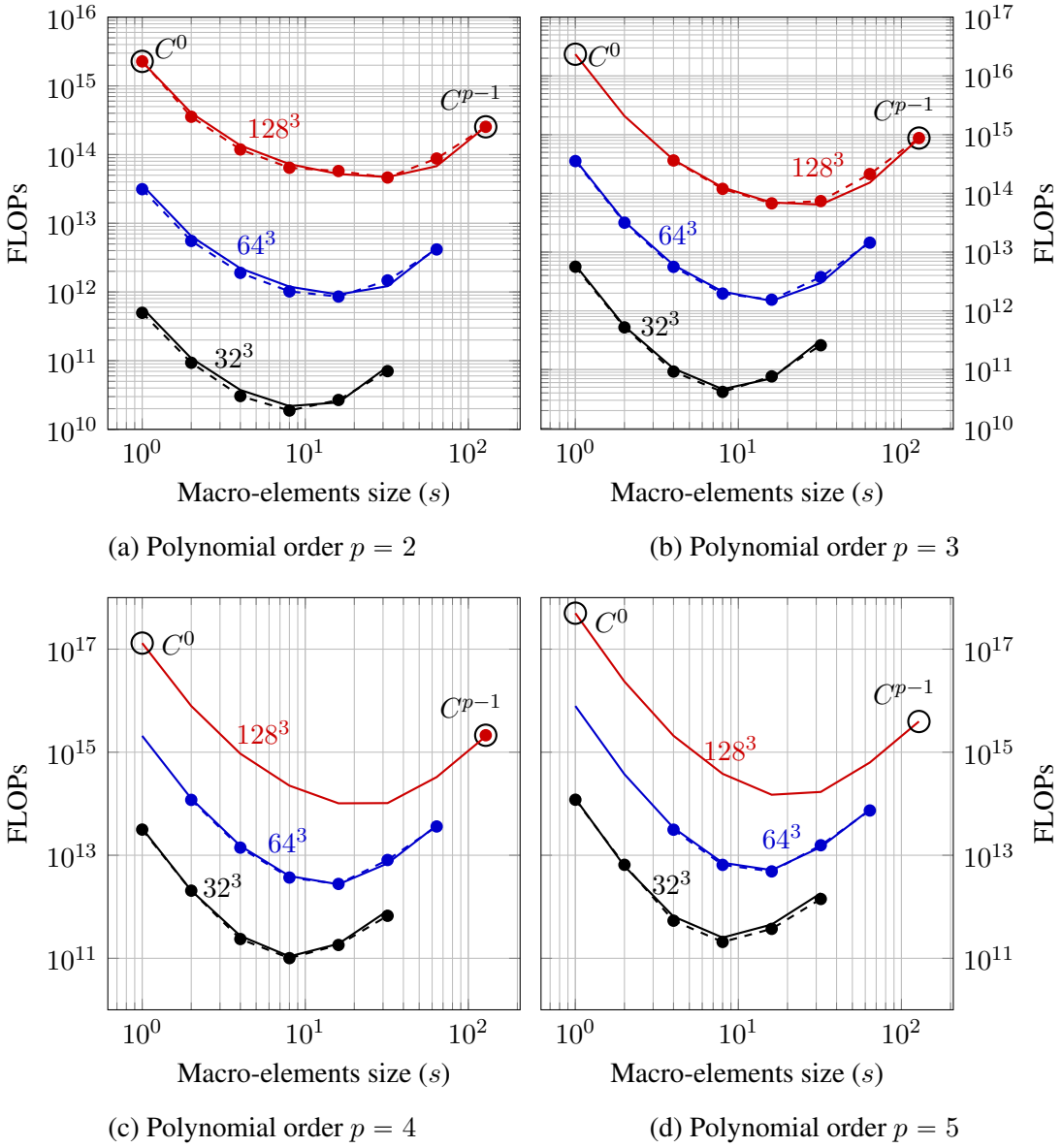
5.1.4.2. Computational times

Figures 5.5 and 5.7 provide the solution times for the model problem in 2D and 3D, respectively. The plots present the solution time with respect to the macro-element size (s). Table 5.2 quantifies the computational times for C^0 FEA, C^{p-1} IGA and the optimal case of rIGA for every polynomial order in a fine mesh.

The computational time required to eliminate the DoF mostly consists of the time spent in performing the FLOPs. Due to this, the solution times show a closer behavior to the number of FLOPs. The rIGA optimal discretization involves a reduction factor in computational time of approximately p^2 with respect to C^{p-1} IGA. For instance, when discretizing the 2D model problem with $N_{\text{elem}} = 2048^2$, a polynomial degree $p = 5$ and macro-elements with size $s = 16$, the reduction is of a factor of approximately 22.

The system that results from discretizing the model problem with $N_e = 2048^2$ and a polynomial order $p = 9$ (Figure 5.6) reports the maximum gain in solution time for 2D, corresponding to a time factor of approximately 37.2 when using MUMPS. The rIGA optimal system requires almost 3 minutes to be solved, while the system obtained from C^{p-1} IGA is solved in approximately 2 hours. For this problem, the rIGA cases with macro-element sizes $s \leqslant 32$ are computed with PARDISO due to PETSc+MUMPS restrictions. PARDISO shows larger computational costs than MUMPS. For instance, PARDISO spends 150 min to solve the C^{p-1} IGA case instead of the 110 min required by MUMPS.

In 3D, the system discretized with $N_{\text{elem}} = 128^3$ and polynomial order $p = 3$ reports a reduction factor of 13.69, Thus, the problem is solved in 1 hour with rIGA instead of the 15 hours required when using C^{p-1} IGA. This is the maximum reproducible gain we obtained in



(a) Polynomial order $p = 2$

(b) Polynomial order $p = 3$

(c) Polynomial order $p = 4$

(d) Polynomial order $p = 5$

Figure 5.4.: Number of FLOPs required to eliminate the degrees of freedom in the 3D model problem (when using the multifrontal direct solver). The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

3D, since the C^{p-1} cases solved with a mesh size $N_{\text{elem}} = 128^3$ and polynomial orders $p = 4$ and $p = 5$ could not be resolved due to memory limitations. Nonetheless, theoretical estimates clearly indicate that gains associated to rIGA vastly increase as we increment p.

Mesh size	Method	Polynomial degree				
		2	3	4	5	9
1024^2	C^{p-1} IGA	1.10e+2	3.20e+2	6.90e+2	1.30e+3	6.70e+3
	C^0 FEA	1.70e+2	4.20e+2	***	***	***
	rIGA (Optimal case)	3.20e+1	3.70e+1	4.60e+1	5.90e+1	1.80e+2
	Gain	3.44	8.65	15	22.03	37.2
128^3	C^{p-1} IGA	1.40e+4	5.40e+4	2.1e+03	4.2e+03	***
	C^0 FEA	***	***	***	***	***
	rIGA (Optimal case)	3.40e+3	3.90e+03	1.9e+02	3.2e+02	***
	Gain	4.28	13.69	11.32	13.08	***

Note: The asterisks ($***$) reflect that the computation exceeds the maximum available physical memory, thus the solution failed (out of memory).
Blue stands for a mesh size $N_{\text{elem}} = 64^3$.

Table 5.2.: Computational time (in seconds) for the model problem discretized with the asymptotic mesh and maximum gain of the optimal case with respect to the C^{p-1} IGA discretization.

5.1.4.3. Memory requirements

Figures 5.8 and 5.9 show the memory requirements (in Mbytes) for 2D and 3D, respectively. We display the memory usage reported by the multifrontal solver MUMPS along with the theoretical estimation computed as

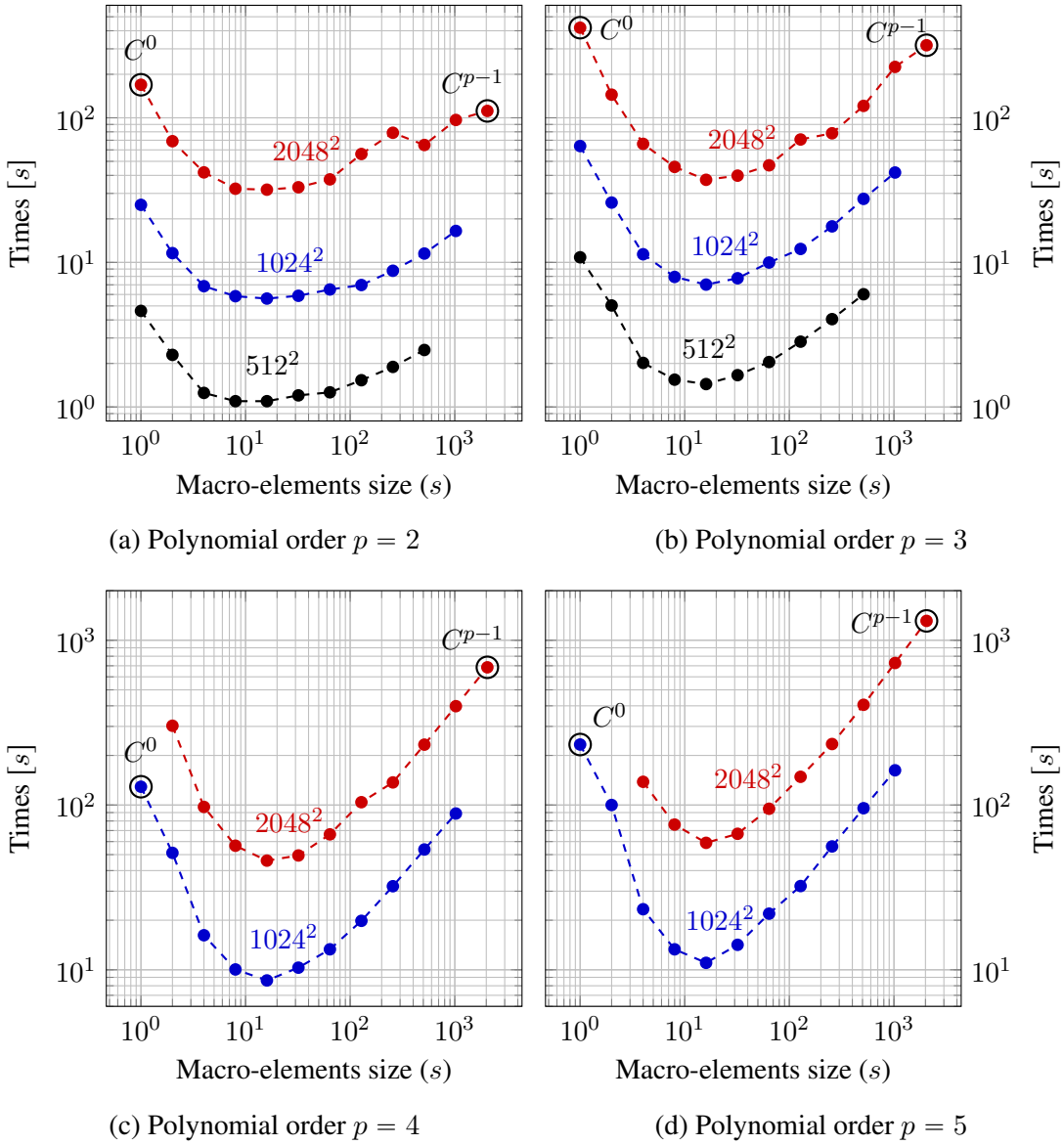
$$2D: \quad \chi = \mathcal{O}\left(\left(\frac{3}{2}\log_2\left(n_{\text{elem}}\right)^2 p^2 + \frac{3}{2}\log_2\left(2^{-i}\right)\right) n^2\right) + L.O.T.\,, \quad [Mbytes]$$

$$3D: \quad \chi = \mathcal{O}\left(\left(\frac{7}{2}\left(2^{-i}\right) p^2 + \frac{7}{2}\left(1-2^{-i}\right)\right) n^4\right) + L.O.T., \quad [Mbytes]$$

which are derived in Appendix A.

In 2D, the maximum reduction in memory usage among those considered in Figure 5.8 corresponds to the case with $N_{\text{elem}} = 2048^2$ and polynomial order $p = 5$. The factor of memory reduction is 4. In 3D, the system discretized using $N_{\text{elem}} = 128^3$ and polynomial orders $p = 5$ reports the maximum reduction in memory requirements. In this case, the theoretical factor is 2.9. Systems discretized with higher polynomial orders involve larger reduction of memory usage.

The discrepancies observed in 3D, for high p and low n, between the theoretical estimates and the recorded data for memory usage are due to the exclusion of lower order terms from the theoretical estimates. Such lower order terms (in the 3D memory estimates) are only one power of n smaller than the dominant cost, but they are multiplied by a larger power of p. Thus, they become dominant on the pre-asymptotic regime (low n and large p). This situation does not



(a) Polynomial order $p = 2$

(b) Polynomial order $p = 3$

(c) Polynomial order $p = 4$

(d) Polynomial order $p = 5$

Figure 5.5.: Computational time (in seconds) to factorize the 2D model problem (when using the multifrontal direct solver).

occur in the FLOPs estimates, because for that case, the dominant cost is n^3 times larger than lower order terms, and thus, estimates rapidly arrive at the asymptotic regime.

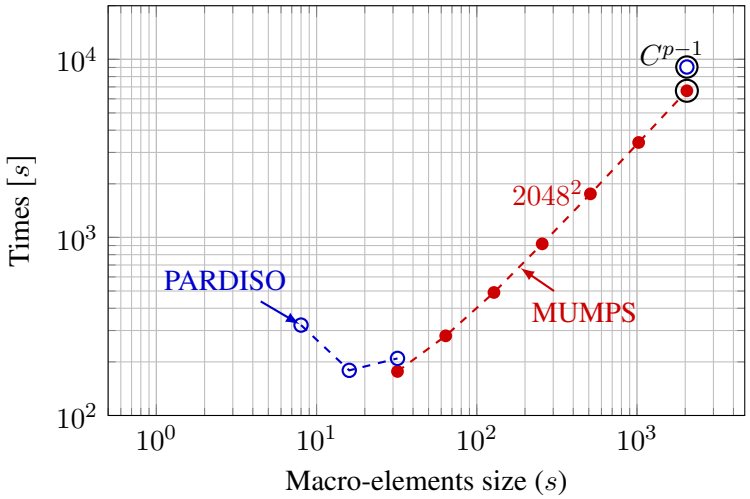
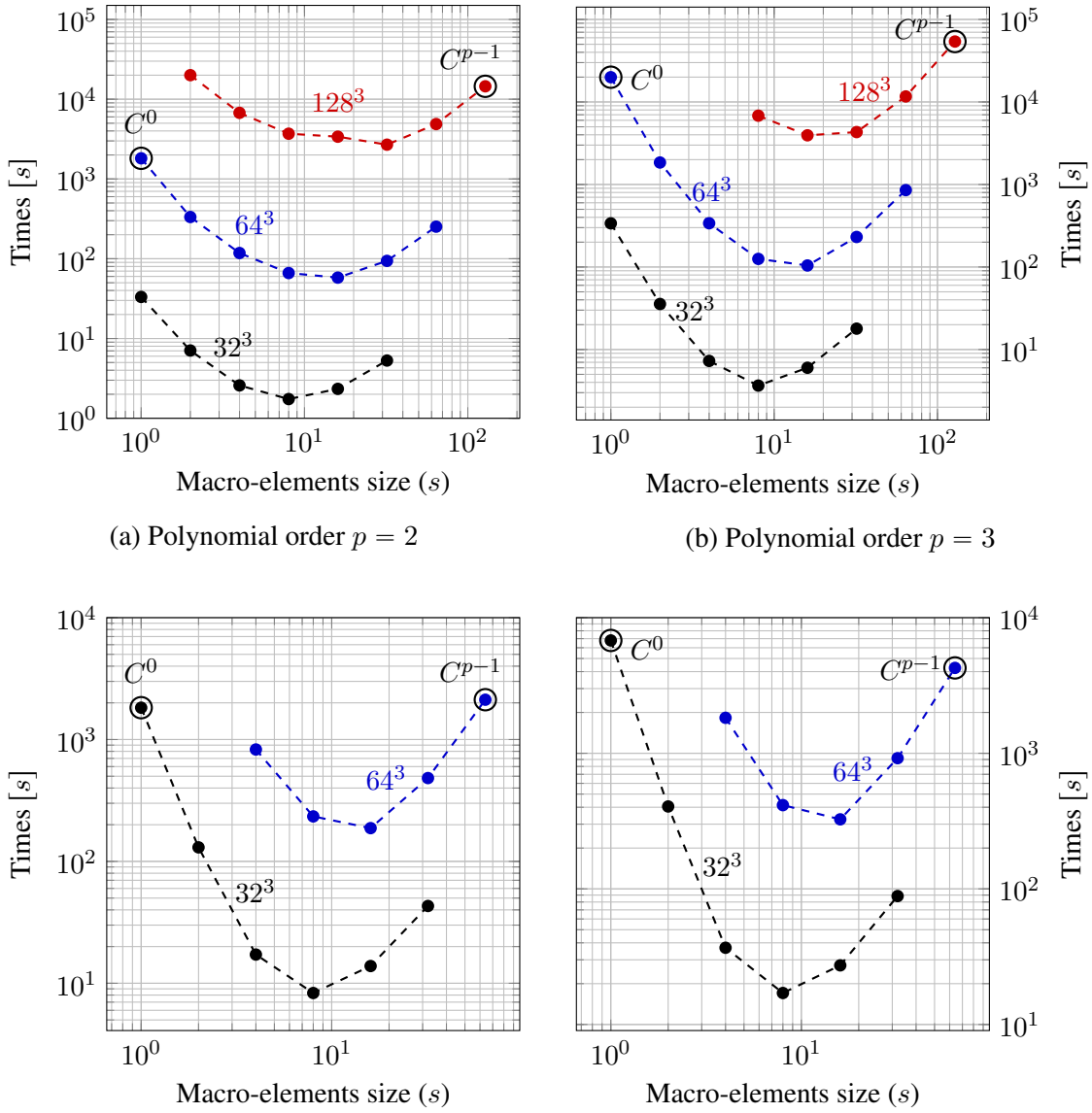


Figure 5.6.: Computational time (in seconds) to factorize the 2D model problem discretized with $N_{\text{elem}} = 2048^2$ elements and $p = 9$. The problem is solved with MUMPS (-●-) and PARDISO (-⊖-) solvers.



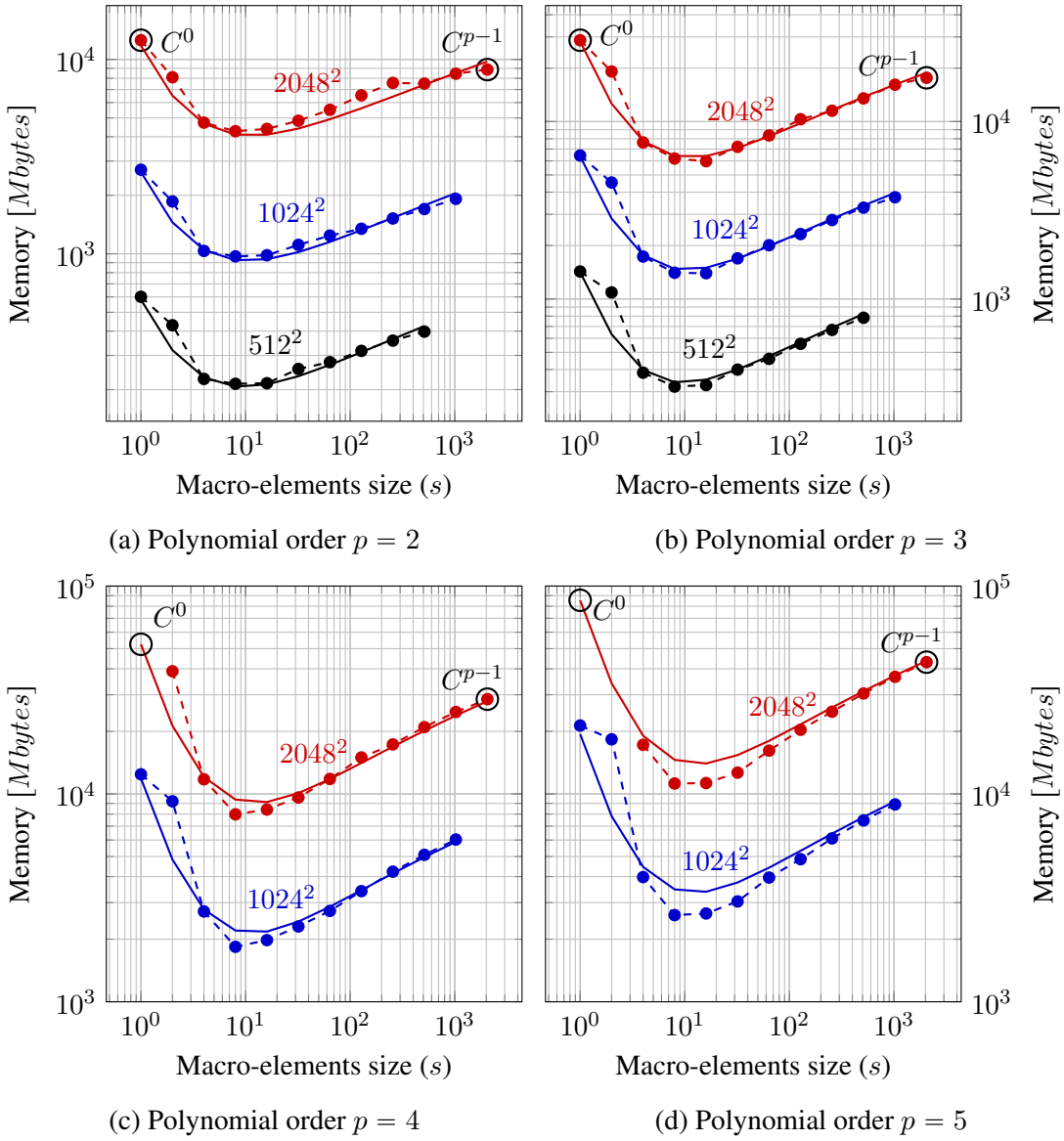
(a) Polynomial order $p = 2$

(b) Polynomial order $p = 3$

(c) Polynomial order $p = 4$

(d) Polynomial order $p = 5$

Figure 5.7.: Computational time (in seconds) for the 3D model problem (when using the multi-frontal direct solver).



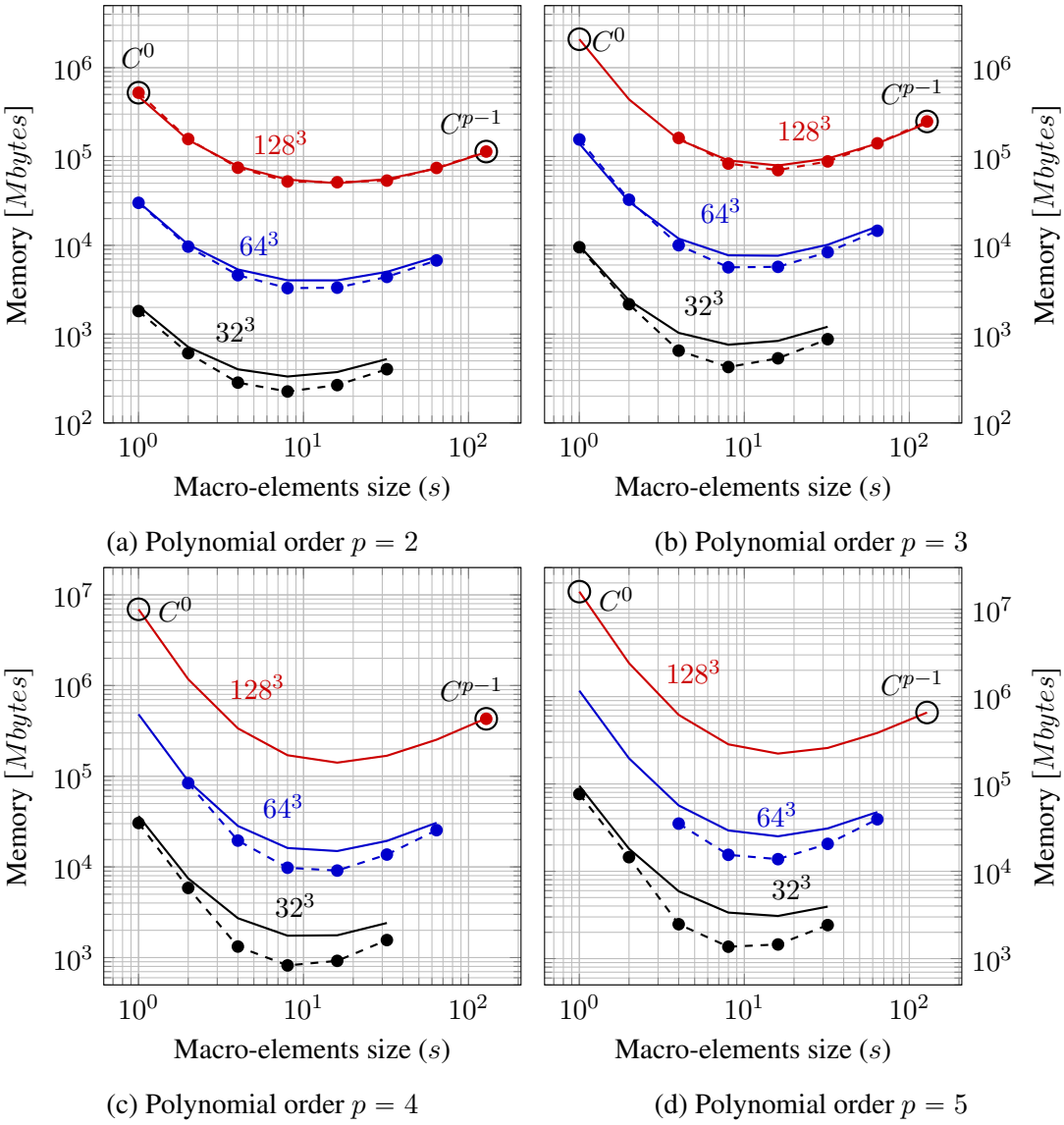
(a) Polynomial order $p = 2$

(b) Polynomial order $p = 3$

(c) Polynomial order $p = 4$

(d) Polynomial order $p = 5$

Figure 5.8.: Memory requirements for the factorization of the 2D model problem (when using the multifrontal direct solver). The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.



(a) Polynomial order $p = 2$

(b) Polynomial order $p = 3$

(c) Polynomial order $p = 4$

(d) Polynomial order $p = 5$

Figure 5.9.: Memory requirements for the factorization of the 3D model problem (when using the multifrontal direct solver). The dashed lines with rounded markers (-●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

5.2. OrIGA for direct solvers

We now consider the case of OrIGA. To analyze the performance of this approach, we use a model problem based on the Laplace equation (5.1) on a 2D domain (Figure 5.1a). This model problem is the same as the one presented in subsection 5.1.1. In our examples, we consider three mesh sizes ($N_{\text{elem}} = 512^2$, 1024^2 and 2048^2) and three polynomial degrees ($p = 5$, 7, and 9).

5.2.1. Numerical results

5.2.1.1. Continuity vectors

Table 5.3 provides the rIGA and OrIGA continuity vectors for various mesh sizes and polynomial degrees. These vectors show that the optimal size of highly continuous macro-elements is almost independent of the mesh size. Moreover, the optimal macro-element size is either 8^2 or 16^2 for both rIGA and OrIGA. In case of OrIGA, the macro-elements involve strong interconnection between some subdomains due to the use of hyperplanes with higher continuity than C^0.

N_{elem}	p	rIGA Continuity	OrIGA Continuity	
512^2	5	$[0.0.0.0.0.0.4.4.4.]$	$[0.0.0.0.0.0.4.4.4.]\|_x$	$[0.0.0.0.0.2.4.4.4.]\|_y$
	7	$[0.0.0.0.0.0.6.6.6.]$	$[0.0.0.0.0.0.6.6.6.]\|_x$	$[0.0.0.0.0.2.6.6.6.]\|_y$
	9	$[0.0.0.0.0.0.8.8.8.]$	$[0.0.0.0.0.0.8.8.8.]\|_x$	$[0.0.0.0.0.1.8.8.8.]\|_y$
1024^2	5	$[0.0.0.0.0.0.0.4.4.4.]$	$[0.0.0.0.0.0.0.4.4.4.]\|_x$	$[0.0.0.0.0.1.3.4.4.4.]\|_y$
	7	$[0.0.0.0.0.0.0.6.6.6.]$	$[0.0.0.0.0.0.0.6.6.6.]\|_x$	$[0.0.0.0.0.1.2.6.6.6.]\|_y$
	9	$[0.0.0.0.0.0.8.8.8.8.]$	$[0.0.0.0.0.0.0.8.8.8.]\|_x$	$[0.0.0.0.0.1.2.8.8.8.]\|_y$
2048^2	5	$[0.0.0.0.0.0.0.4.4.4.4.]$	$[0.0.0.0.0.0.0.1.4.4.4.]\|_x$	$[0.0.0.0.0.0.2.4.4.4.4.]\|_y$
	7	$[0.0.0.0.0.0.0.0.6.6.6.]$	$[0.0.0.0.0.0.0.1.6.6.6.]\|_x$	$[0.0.0.0.0.0.1.4.6.6.6.]\|_y$
	9	$[0.0.0.0.0.0.0.8.8.8.8.]$	$[0.0.0.0.0.0.0.0.8.8.8.]\|_x$	$[0.0.0.0.0.0.1.3.8.8.8.]\|_y$

Table 5.3.: Continuity vectors of rIGA and OrIGA in 2D for various degrees and mesh sizes. OrIGA continuity vectors vary from their corresponding rIGA counterparts, at most, by two values (coordinates) and this variance appears nearby the discontinuity jump of rIGA.

5.2.1.2. FLOPs

Table 5.4 shows the FLOPs estimates of C^0 FEA, C^{p-1} IGA, rIGA, and OrIGA for various mesh sizes. We see that both rIGA and OrIGA significantly outperform FEA and IGA. Moreover, OrIGA shows the expected superior results.

Table 5.5 shows the actual number of FLOPs and computational times using MUMPS. The FLOPs estimates shown in Table 5.4 agree qualitatively (up to a constant) with the numerical

Polynomial degree	Method	$N_{\text{elem}} = 512^2$	$N_{\text{elem}} = 1024^2$	$N_{\text{elem}} = 2048^2$
5	C^0 FEA	3.46e+11	2.73e+12	2.16e+13
	C^{p-1} IGA	3.75e+11	2.87e+12	2.23e+13
	rIGA	1.33e+10	8.97e+10	6.27e+11
	OrIGA	1.26e+10	7.88e+10	5.05e+11
7	C^0 FEA	1.00e+12	7.69e+12	6.02e+13
	C^{p-1} IGA	1.09e+12	8.13e+12	6.23e+13
	rIGA	3.07e+10	1.81e+11	1.18e+12
	OrIGA	2.99e+10	1.67e+11	9.98e+11
9	C^0 FEA	2.36e+12	1.73e+13	1.32e+14
	C^{p-1} IGA	2.32e+12	1.74e+13	1.33e+14
	rIGA	7.03e+10	3.47e+11	1.68e+12
	OrIGA	4.91e+10	2.69e+11	1.61e+12

Table 5.4.: Estimates of the number of FLOPs required by LU factorization (when using multifrontal direct solver). Values obtained with Equation (4.18), when applied to the tested discretizations with various degrees and mesh sizes.

Polynomial degree	Method	$N_{\text{elem}} = 512^2$		$N_{\text{elem}} = 1024^2$		$N_{\text{elem}} = 2048^2$	
		FLOPs	time	FLOPs	time	FLOPs	time
5	C^0 FEA	3.48e+11	36.2	2.94e+12	232.9	***	***
	C^{p-1} IGA	3.01e+11	22.5	2.50e+12	162.5	2.21e+13	1310.9
	rIGA	1.38e+10	2.8	1.01e+11	13.7	5.31e+11	56.5
	OrIGA	1.36e+10	2.4	8.29e+10	12.5	5.59e+11	59.7
7	C^0 FEA	9.71e+11	86.2	***	***	***	***
	C^{p-1} IGA	8.31e+11	53.9	6.94e+12	405.9	5.81e+13	3204.3
	rIGA	2.96e+10	5.9	1.91e+11	24.9	1.32e+12	129.7
	OrIGA	2.88e+10	5.1	1.71e+11	21.7	1.06e+12	104.7
9	C^0 FEA	***	***	***	***	***	***
	C^{p-1} IGA	1.65e+12	102.1	1.41e+13	820.1	1.19e+14	6657.8
	rIGA	8.16e+10	10.1	3.81e+11	44.0	***	***
	OrIGA	6.99e+10	8.6	3.24e+11	36.7	***	***

Note: The asterisks ($***$) reflect that the computation was not accomplished due to memory limitations (out of memory).

Table 5.5.: Computed number of FLOPs and computational times (in seconds) required by MUMPS to factorize the 2D problem.

results. In addition, the numerical results confirm the superiority of OrIGA. For instance, for $p = 7$ and $N_{\text{elem}} = 2048^2$, OrIGA requires 55 times less number of FLOPs than C^{p-1} IGA. Additionally, OrIGA is 25% cheaper than rIGA.

In the particular case of a problem discretized with a mesh size $N_{\text{elem}} = 2048^2$ and $p = 5$, OrIGA becomes computationally more expensive than rIGA. This occurs because the continuity vectors provided by the search algorithm to solve this case are not globally optimal, since estimates used to find such global optima are only quasi-optimal. An improved search is necessary in this case.

Summarizing, for direct solvers, rIGA delivers a reduction factor of the computational cost for solving Laplace based problems of up to p^2 (in terms of FLOPs) with respect to C^{p-1} IGA, being p the polynomial degree. We can solve other scalar problems with rIGA, e.g., Helmholtz and diffusion advection reaction problems. To compute the solution of those problems, we use Galerkin formulations that satisfy the required stability criterion, and discretizations that are similar to those for the Laplace based problems. Due to this, we achieve the same gain factors in computational cost than the ones observed for Laplace problems. Moreover, the reduction of continuity enriches the space solutions, thereby improving the best approximation error.

For Multi-field problems (vectorial problems) discretized using $\mathbf{H}^1 = (H^1)^{d_f}$ spaces (being d_f the number of vector fields), rIGA delivers a similar computational cost reduction factor than for the scalar Laplace model problems solved using a H^1 space. In here, we denote H^1 to the space of functions in L^2 whose first-order derivates belongs to L^2. In those multi-field problems, the cost of factorization for IGA and rIGA is given by

$$\text{IGA:} \qquad \theta^{\mathbf{H}^1} = \mathcal{O}\left(d_f^3 \left(n_{\text{elem}} + p\right)^{3(d-1)} p^3\right), \qquad \text{[FLOPs]}$$

$$\text{rIGA:} \quad \theta^{\mathbf{H}^1} = \left(2^{(3-2d)\ell} \cdot \mathcal{O}\left(d_f^3 n^{3(d-1)} p^3\right) + \mathcal{O}\left(d_f^3 n^{3(d-1)}\right)\right), \quad \text{[FLOPs]}$$

where $n = (n_{\text{elem}} + p + (2^\ell - 1)(p-1))$. The factorization cost is approximately d_f^3 times more expensive than for a scalar Laplace problem (see Equations 4.5 and 4.9). In addition, the reduction factor (between rIGA and IGA) of the computational cost is of up to p^2.

For multi-field problems solved using $\mathbf{H}(div)$ or $\mathbf{H}(curl)$ spaces, we may obtain slightly smaller reduction factors than for cases using $\mathbf{H}^1$ spaces. $\mathbf{H}(div)$ and $\mathbf{H}(curl)$ spaces involve different polynomial degrees per spatial direction. For instance, the $\mathbf{H}(div)$ and $\mathbf{H}(curl)$ spaces in 3D

$$\mathbf{H}(div): \qquad \mathcal{S}_{k+1,k,k}^{p+1,p,p} \times \mathcal{S}_{k,k+1,k}^{p,p+1,p} \times \mathcal{S}_{k,k,k+1}^{p,p,p+1},$$

$$\mathbf{H}(curl): \quad \mathcal{S}_{k,k,k+1}^{p,p+1,p+1} \times \mathcal{S}_{k+1,k,k+1}^{p+1,p,p+1} \times \mathcal{S}_{k+1,k+1,k}^{p+1,p+1,p},$$

where k is the continuity degree. By reducing the continuity in the same degree (not to the same order), we obtain thicker separators at some partition levels than when using $\mathbf{H}^1$-discretizations. Thus, the reduction in computational cost are expected to be slightly smaller (by a factor independent of the discretization).

The cost estimate in term of FLOPs for $\mathbf{H}(div)$ and $\mathbf{H}(curl)$ discretizations in C^{p-1} IGA are

$$\theta_{\text{IGA}}^{\mathbf{H}(div)} = \mathcal{O}\left(\left((n_{\text{elem}}+p)^{(d-1)}\,(dp+1)+(d-1)\,(n_{\text{elem}}+p)^{(d-2)}\,p\right)^3\right),$$

$$\theta_{\text{IGA}}^{\mathbf{H}(curl)} = \mathcal{O}\Big(\Big((n_{\text{elem}}+p)^{(d-1)}\,(d(p+1)-1) + (d-1)\,(n_{\text{elem}}+p)^{(d-2)}\,((d-1)\,(p+1)-1) + p\,(d-2)\Big)^3\Big).$$

This cost becomes

$$\theta_{\text{rIGA}}^{\mathbf{H}(div)} = \left(2^{(3-2d)\ell}\cdot\mathcal{O}\left(\left(n^{(d-1)}(d\,p+1)+(d-1)n^{(d-2)}p\right)^3\right) + \mathcal{O}\left(\left(n^{(d-1)}(d+1)+(d-1)n^{(d-2)}\right)^3\right)\right),$$

$$\theta_{\text{rIGA}}^{\mathbf{H}(curl)} = \left(2^{(3-2d)\ell}\cdot\mathcal{O}\Big(\Big(n^{(d-1)}\,(d(p+1)-1) + (d-1)\,n^{(d-2)}\,((d-1)\,(p+1)-1) + p\,(d-2)\Big)^3\Big) + \mathcal{O}\Big(\Big(n^{(d-1)}\,(2d-1) + (d-1)\,n^{(d-2)}\,(2\,(d-1)-1) + (d-2)\Big)^3\Big)\right),$$

when reducing the continuity along all the separators on ℓ partition levels (rIGA). Lastly, the reduction factor of the computational cost between rIGA and IGA is of $\mathcal{O}(p^2)$ once we reach the asymptotic regime. A detailed explanation of how to perform the continuity reduction on those spaces is provided in Chapter 6.

5.3. rIGA for iterative solvers

To analyze the computational cost when using the hybrid solver strategy in 2D, we focus in both, the static condensation of the macro-elements interior DoF and the solution of the skeleton system using the preconditioned iterative solver. The remaining operations only contribute with lower order terms ($L.O.T.$). We report the FLOPs and computational times (in seconds) with respect to the one-dimensional size (s) of the macro-elements. In here, we only focus on the cases which deliver the maximum reduction in solution time. Therefore, we present the numerical results for C^0 FEA, C^{p-1} IGA and the optimal cases of rIGA.

After that, we present an analysis of the cost when using the hybrid solver strategy in 3D. For this case, we compare the number of NZ entries of the matrix A in the global system with that of the skeleton matrix A_{skl} resulting after performing static condensation over the macro-elements. In 3D, the total cost is ruled by the iterative solver. This occurs because the matrices A_{skl} involve a huge number of NZ entries. Therefore, the number of NZ entries becomes a suitable estimate of the computational cost.

5.3.1. Model problem

We use the Poisson model problem presented in Equation (5.2) to analyze the performance of the hybrid solver strategy. In particular, we study the impact of using refined Isogeometric Analysis (rIGA) with a preconditioned CG iterative solver in the solution cost.

$$\begin{cases} \text{Find } u \text{ such that:} & \\ \nabla \cdot (\nabla u) = f(\alpha) & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{5.2}$$

where $\Omega = (0,1)^d$ and $f(\alpha) = d\,\alpha^2\pi^2 \prod_{i=1}^{d} \sin(\alpha\,\pi\,\mathbf{x}(i))$, being $\mathbf{x} = (x, y, z)$ and d the spatial dimension. We use a different value of α for each polynomial degree to have approximately the same $||e_D|_{\text{IGA}}||_1$ in all the examples. The problem domain for 2D is illustrated in Figure 5.10.

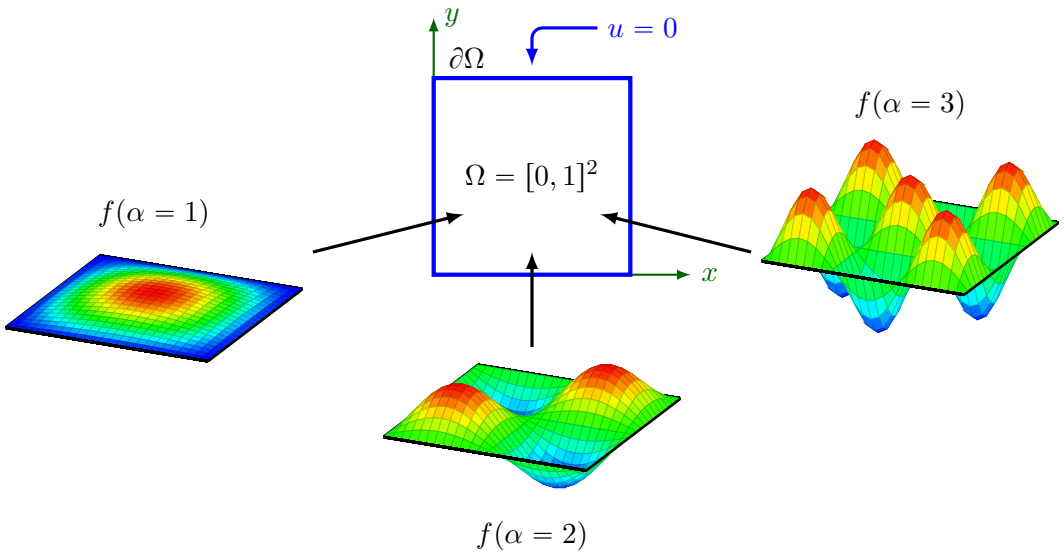


Figure 5.10.: Poisson model problem domain $\left(\Omega = (0,1)^2\right)$.

5.3.2. Implementation details

We build the model problem using unmapped B-splines and assuming a unitary domain. This domain is defined by the tensor product of the unmapped B-splines resulting in a regular and

structured mesh. The same number of elements in all spatial dimensions results in uniform macro-elements after performing the domain partitioning. We discretize the model problem with a mesh of 2048^2 and 4096^2 elements, and polynomial degrees $p = 2$, 3, 4, 5 and 6. The polynomial degree is kept constant on the entire mesh.

For every mesh size, we consider several cases. Each case splits the mesh into a particular number of macro-elements using C^0-hyperplanes. The first one corresponds to traditional C^{p-1} IGA with no reduction of continuity, while the last case involves a total reduction of continuity which results in a traditional C^0 FEA.

In this implementation, we set the stopping criteria of the CG iterative solver as:

$$||e||_1 = ||e_I + e_D||_1 \leqslant \xi, \tag{5.3}$$

where e is the total error consisting of the sum of the error of the iterative solver (e_I) and the discretization error (e_D). We select $\xi = 2||e_D|_{\text{IGA}}||_1$, where $||e_D|_{\text{IGA}}||_1$ is the H^1-norm of the discretization error (e_D) resulting from solving the model problem with C^{p-1} IGA. We pick ξ as two times the value of $||e_D|_{\text{IGA}}||_1$. This is small enough to allow the iterative solver to reach a stable rate of convergence.

5.3.3. Fit of estimates

We fit the theoretical FLOPs estimates with the numerical results using three constants, namely *A*, *B* and *C*. The theoretical estimate of the number of FLOPs required to solve the problem is given by:

$$\theta_{\text{rIGA}}|_{\text{CG+ILU}} = \underbrace{A\,\theta_{SC}}_{\text{Static Condensation}} + \underbrace{B\,\theta_{pre}}_{\text{Preconditioning}} + \underbrace{N_{ite}\,(C\,\theta_{it})}_{\text{Iterative solver}}, \tag{5.4}$$

where N_{ite} is the number of iterations that the solver requires to converge. Table 5.6 lists the values of the constants *A*, *B* and *C*.

Constants	Polynomial degree p				
	2	3	4	5	6
A	28	22	18	16	16
B	0.68				
C	1				

Table 5.6.: Fitting constants computed for every polynomial degree in 2D.

Constant *A* captures the total number of FLOPs employed by MUMPS to reduce the system matrices and to form the Schur complements of each macro-element [27]. In particular, *A* accounts for the constant accompanying the leading term of the LU factorization cost performed by LAPACK, and the contribution of the matrix-matrix and matrix-vector multiplication and subtraction required to form the Schur matrices. For small p, the contribution of forming those matrices is significant compared with the cost of LU factorization, influencing the value of constant *A*. However, the contribution of those operations becomes smaller as the polynomial

degree grows ($p \gg 1$). Constant B corrects the overestimation in the number of operations of the truncated Gaussian elimination that occurs at some non-zero entries of the skeleton mesh. This overestimate is related to the fact that θ_{pre} estimates the cost of preconditioning by assuming a periodic mesh. Thus, θ_{pre} counts additional operations at some non-zero entries. Finally, θ_{it} exactly predicts the number of FLOPs required per iteration by the iterative solver to converge. Therefore, constant C is one. We take N_{ite} equal to the number of iterations that results from the numerical experiments.

5.3.4. 2D numerical results

5.3.4.1. Cost of static condensation

Figures 5.11 and 5.12 illustrate the number of FLOPs as well as the average computational time (in seconds) required to eliminate the internal DoF for various macro-element sizes.

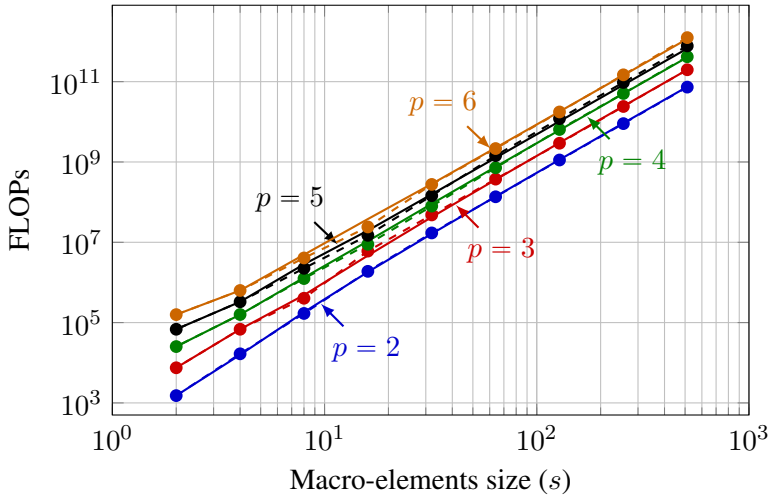


Figure 5.11.: Number of FLOPs required to eliminate the interior DoF in a single macro-element. The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

When the macro-element size is large ($s \gg 1$), the computational time required to eliminate the internal degrees of freedom mainly consists of the time spent in performing FLOPs. For instance, when we increase the macro-element size from 64^2 to 128^2, the computational time (Figure 5.12) grows linearly with the number of FLOPs (Figure 5.11). However, when the macro-element size becomes closer to one ($s \to 1$), many other factors affect the computational cost, e.g., the bandwidth limit of global memory access [51]. Indeed, the procedure of computing LU factors for multiple small sparse systems becomes memory bound. Thus, in such limit, the cost of memory access becomes dominant, increasing the factorization time as shown in Figure 5.12.

The cost to build the skeleton system (perform the static condensation) is equal to the sum of the cost of performing the partial factorization over *all* the macro-elements. Figure 5.13

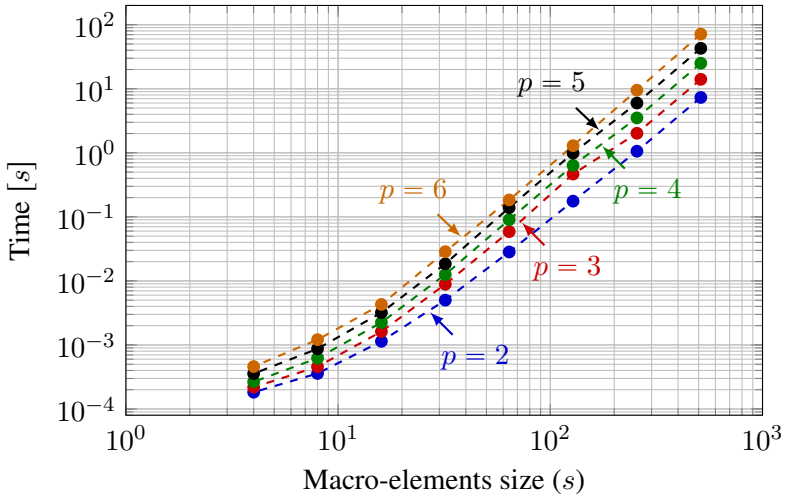


Figure 5.12.: Computational time required to eliminate the interior DoF in a single macro-element.

illustrates the number of FLOPs and Figure 5.14 the computational time required to perform the static condensation in *all* the macro-elements.

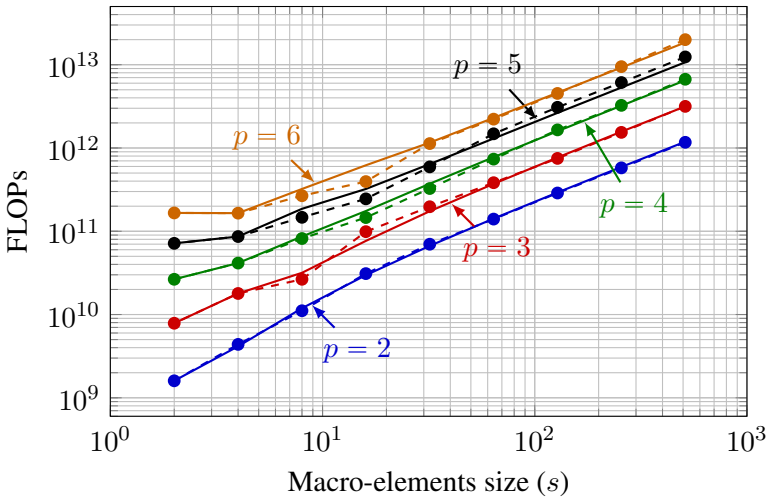


Figure 5.13.: Number of FLOPs required to eliminate the interior DoF in *all* macro-elements for a problem with a mesh size of $N_{\text{elem}} = 2048^2$. The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

The rate of change of the number of FLOPs required to eliminate the interior degrees of

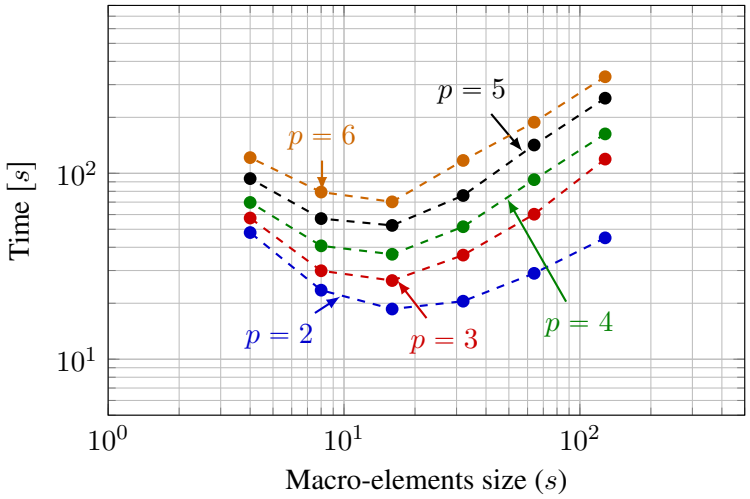


Figure 5.14.: Computational time required to eliminate the interior DoF in *all* macro-elements for a problem with a mesh size of $N_{\text{elem}} = 2048^2$.

freedom in *all* macro-element with respect to those on the element interfaces is

$$\mathcal{O}\left(\frac{(\eta_{\text{m-e}}\,\theta_{SC})_\ell}{(\eta_{\text{m-e}}\,\theta_{SC})_{\ell+1}}\right) \sim \mathcal{O}\left(\left(\frac{(n_{\text{elem}}/s)^2_\ell}{(n_{\text{elem}}/s)^2_{\ell+1}}\right)\left(\frac{(N^\circ_{dof})^\beta_\ell}{(N^\circ_{dof})^\beta_{\ell+1}}\right)\right)$$
$$\sim \mathcal{O}\left(\left(\frac{s^2_{\ell+1}}{s^2_\ell}\right)\left(\frac{(s+p-2)^{2\beta}_\ell}{(s+p-2)^{2\beta}_{\ell+1}}\right)\right),$$

where ℓ refers to the ℓ-th partition level. For macro-element sizes larger than the polynomial degree ($s \gg p$), $\beta = 3/2$ (sparse matrix), the rate of change becomes approximately $\mathcal{O}\left(s_\ell/s_{\ell+1}\right)$. This explains the linear reduction of the number of FLOPs as the macro-elements become smaller. Moreover, as soon as the macro-element size becomes of the same order of the polynomial degree ($s \sim p$), β becomes three (dense matrix) and the rate of change starts to behaves approximately as $\mathcal{O}\left(s^2_{\ell+1}/s^2_\ell\right)$. This explains the growth in the number of FLOPs observed when the macro-element size is close to one (Figure 5.13).

In addition to the numerical factorization, the direct methods perform an analysis phase. This phase involves the reordering of the matrix as well as additional operations to allocate memory. Assuming that these operations can be computed once and then reused by *all* macro-elements, we considered this cost as $L.O.T.$ and we do not include it in the total computational cost.

5.3.4.2. Cost of preconditioner set-up

Before solving the skeleton system, we use the ILU factorization strategy with zero fill-ins to build the preconditioner matrix. Figures 5.15 and 5.16 illustrates the number of FLOPs and the computational time (in seconds) required to construct P, respectively. In those Figures, the points on the right (located over the vertical dashed line) correspond to the number of FLOPs and computational times required to build the preconditioner matrix for the IGA cases.

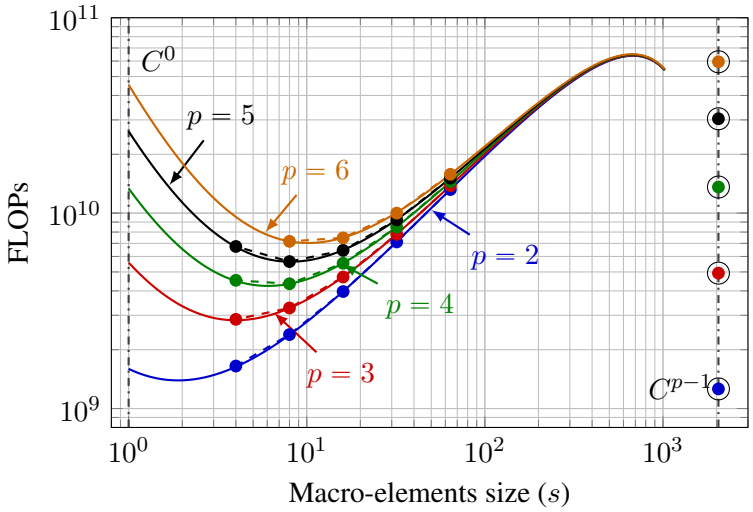


Figure 5.15.: Number of FLOPs required to build the preconditioner matrix P for the skeleton system when solving the 2D Poisson model problem with a mesh size of $N_{\text{elem}} = 2048^2$. The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

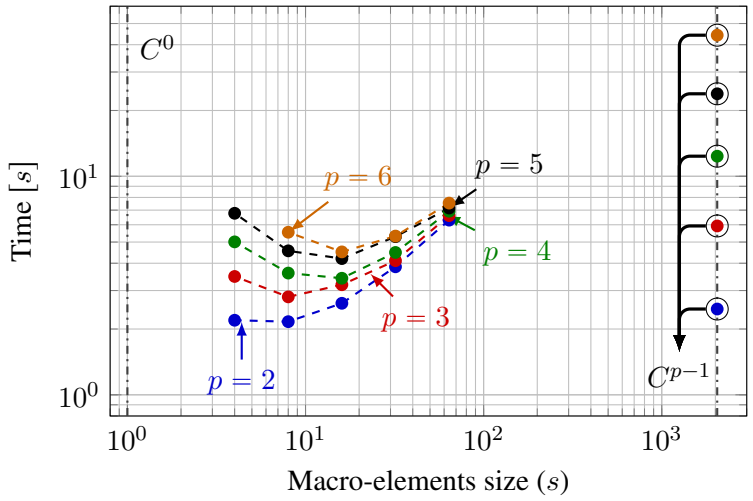


Figure 5.16.: Computational time required to build the preconditioner matrix P for the skeleton system when solving the 2D Poisson model problem with a mesh size of $N_{\text{elem}} = 2048^2$.

The cost to compute the preconditioner matrix becomes almost independent of the order of approximation p for cases with macro-element sizes larger than the polynomial degree ($s \gg p$). For those cases, $\hat{p} \approx s$ and therefore θ_{pre} mostly depends on s, given by Equation 4.36.

We observe a reduction in the cost to build the preconditioner only when using discretizations

with polynomial degrees higher than two ($p > 2$). For instance, the case with 2048^2 elements and $p = 2$ involves no cost reduction, but the case with $p = 7$ involves a reduction factor of approximately 13 in both the number of FLOPs and computational time.

5.3.4.3. Cost of solving the skeleton system

The cost to solve the skeleton system depends on the number of NZ entries of matrix A_{skl}, as well as the number of iterations required to converge within the desired tolerance (Equation 5.3). Table 5.7 illustrates the value of the H^1-norm of the discretization error resulting from solving the problem with C^{p-1} IGA $\left(||e_D|_{\text{IGA}}||_1\right)$ for all polynomial degrees and a mesh size of 2048^2 elements.

n_{elem}	Polynomial degree p				
	2	3	4	5	6
2048	1.95e-7	2.88e-8	1.59e-8	1.30e-8	1.08e-8

Table 5.7.: H^1-norm of the discretization errors (e_D) resulting from approximating the 2D Poisson problem solution with C^{p-1} IGA, a mesh size of 2048^2 elements and several polynomial degrees.

Matrix A_{skl} has approximately $7(s + p)$ NZ entries per column instead of the $(2p + 1)^2$ in matrix A. This implies that, in most cases, matrix A_{skl} is denser than matrix A. However, matrix A_{skl} is significantly smaller than A. This results in a lower total number of NZ entries as shown in Table 5.8. In there we show that the number of NZ entries of matrix A_{skl} (from the second to the last row in the table) is lower than that of matrix A (first row in the table).

n_{elem}	s	Polynomial degree p				
		2	3	4	5	6
2048	2048 (IGA)	1.05e+8	2.06e+8	3.40e+8	5.09e+8	7.11e+8
	64	6.06e+7	6.25e+7	6.44e+7	6.63e+7	6.38e+7
	32	6.21e+7	6.59e+7	6.98e+7	7.39e+7	7.81e+7
	16	6.52e+7	7.32e+7	8.16e+7	9.05e+7	9.99e+7
	8	7.19e+7	8.91e+7	1.08e+8	1.29e+8	1.52e+8

Table 5.8.: Number of non-zero entries of the skeleton systems.

Table 5.9 shows the number of iterations that the CG iterative solver preconditioned with ILU requires to converge to the desired tolerance. The hybrid strategy reduces the number of iterations in most of the cases. This is a result of reducing the system by eliminating the macro-elements internal DoF. In the skeleton problem, the macro-elements communicate only to the neighboring macro-elements. Therefore, the number of iterations required to solve the system is inversely proportional to the size of each macro-element. This explains the growth in the number of iterations that we observe when the macro-element size reduces (the number of macro-elements increases).

n_{elem}	s	Polynomial degree p				
		2	3	4	5	6
2048	2048 (IGA)	194	222	167	118	78
	64	37	38	47	46	52
	32	57	56	53	57	54
	16	84	89	90	80	65
	8	106	150	150	150	106

Table 5.9.: Number of iterations required by the preconditioned iterative solver CG+ILU(0) to converge.

Tables 5.10 and 5.11 show, respectively, the number of FLOPs and the computational time required to solve the reduced system using the preconditioned iterative solver CG+ILU. The reduction in the cost observed when using the hybrid strategy is explained by both, the lower number of NZ entries that the skeleton matrix A_{skl} has, and the reduction in the number of iterations due to performing static condensation at the level of the macro-elements.

n_{elem}	s	Polynomial degree p				
		2	3	4	5	6
2048	2048 (IGA)	8.98e+10	1.92e+11	2.35e+11	2.46e+11	2.26e+11
	64	9.14e+09	9.67e+09	1.23e+10	1.24e+10	1.44e+10
	32	1.44e+10	1.50e+10	1.51e+10	1.72e+10	1.72e+10
	16	2.25e+10	2.67e+10	3.01e+10	2.97e+10	2.66e+10
	8	3.18e+10	5.55e+10	6.72e+10	7.99e+10	6.64e+10

Table 5.10.: Number of FLOPs required to solve the skeleton system using the CG+ILU solver.

n_{elem}	s	Polynomial degree p				
		2	3	4	5	6
2048	2048 (IGA)	63.14	115.10	137.84	149.72	132.72
	64	5.94	6.24	7.96	8.029	9.31
	32	9.24	9.58	9.57	10.91	11.04
	16	14.30	16.89	18.92	18.68	16.71
	8	19.59	34.25	41.30	50.74	42.13

Table 5.11.: Computational time (in seconds) required to solve the skeleton system using the CG+ILU solver.

5.3.4.4. Total cost of the hybrid solver strategy

The total cost required to solve the 2D Poisson model problem (Equation 5.2) comprehends the costs of the static condensation, the preconditioner construction and the computation of the skeleton system solution. Tables 5.12 and 5.13 show, respectively, the number of FLOPs (θ) and the computational time (τ) required by each of the three steps, and the total cost required to solve the 2D model problem.

n_{elem}	p	s	θ_{SC}	θ_{pre}	θ_{it}	θ
2048	2	2048 (IGA)	***	1.26e+09	8.98e+010	9.11e+10
		64	1.40e+11	1.32e+10	9.14e+009	1.62e+11
		32	6.99e+10	7.12e+09	1.44e+010	9.14e+10
		16	3.10e+10	3.97e+09	2.25e+010	5.75e+10
		8	1.11e+10	2.39e+09	3.18e+010	4.53e+10
	3	2048 (IGA)	***	4.93e+09	1.92e+11	1.97e+11
		64	3.83e+11	1.38e+10	9.67e+09	4.07e+11
		32	1.97e+11	7.79e+09	1.50e+10	2.19e+11
		16	9.89e+10	4.71e+09	2.67e+10	1.30e+11
		8	2.65e+10	3.27e+09	5.55e+10	8.52e+10
	4	2048 (IGA)	***	1.36e+10	2.35e+11	2.49e+11
		64	7.35e+11	1.45e+10	1.23e+10	7.62e+11
		32	3.26e+11	8.49e+09	1.51e+10	3.50e+11
		16	1.46e+11	5.54e+09	3.01e+10	1.81e+11
		8	8.18e+10	4.35e+09	6.72e+10	1.53e+11
	5	2048 (IGA)	***	3.04e+10	2.46e+11	2.76e+11
		64	1.48e+12	1.51e+10	1.24e+10	1.50e+12
		32	5.93e+11	9.24e+09	1.72e+10	6.19e+11
		16	2.45e+11	6.45e+09	2.97e+10	2.81e+11
		8	1.47e+11	5.65e+09	7.99e+10	2.33e+11
	6	2048 (IGA)	***	5.95e+10	2.26e+11	2.86e+11
		64	2.22e+12	1.58e+10	1.44e+10	2.25e+12
		32	1.13e+12	1.00e+10	1.72e+10	1.16e+12
		16	3.95e+11	7.47e+09	2.66e+10	4.29e+11
		8	2.67e+11	7.17e+09	6.64e+10	3.40e+11

Table 5.12.: Number of FLOPs required to perform static condensation (θ_{SC}), construct the preconditioner (θ_{pre}), compute the skeleton system solution (θ_{it}) and solve the 2D Poisson model problem (θ).

The total number of FLOPs (θ) shows a similar behavior than θ_{SC} when the macro-elements sizes decrease. No increment in the value of θ is observed even with the growing contribution of θ_{it}. This implies that the predominant factor contributing to the total cost is the static condensation step. This is best observed in cases with a large polynomial degree (Table 5.12). In addition,

n_{elem}	p	s	τ_{SC}	τ_{pre}	τ_{it}	τ
2048	2	2048 (IGA)	***	2.45	63.13	65.58
		64	27.18	6.45	5.94	39.57
		32	19.86	4.06	9.24	33.18
		16	18.40	2.64	14.30	35.34
		8	23.33	2.23	19.59	45.15
	3	2048 (IGA)	***	5.92	115.99	121.92
		64	49.26	6.60	6.24	62.10
		32	34.78	4.18	9.58	48.54
		16	26.26	3.33	16.89	46.49
		8	32.18	3.00	34.24	69.43
	4	2048 (IGA)	***	12.36	137.84	150.20
		64	80.49	6.92	7.96	95.37
		32	49.79	4.47	9.57	63.84
		16	35.88	3.64	18.92	58.45
		8	41.46	3.63	41.30	86.39
	5	2048 (IGA)	***	23.97	149.72	173.69
		64	133.13	7.16	8.03	148.32
		32	75.20	5.29	10.90	91.41
		16	50.37	4.19	18.68	73.24
		8	58.69	4.76	50.74	114.20
	6	2048 (IGA)	***	44.21	132.72	176.94
		64	188.05	7.54	9.31	204.89
		32	117.35	5.32	11.04	133.71
		16	70.50	4.51	16.71	91.73
		8	79.05	5.54	42.13	126.72

Table 5.13.: Computational time (in seconds) required to perform static condensation (τ_{SC}), construct the preconditioner (τ_{pre}), compute the skeleton system solution (τ_{it}) and solve the 2D Poisson model problem (τ).

the numerical results show that by using the hybrid method, we slightly reduce the total number of FLOPs (θ). Particularly, when using a polynomial degree $p = 3$.

Regarding computational times, the static condensation is the factor that contributes most to the total cost. In most cases, the discretization that delivers the lowest computational time is the one that involves the minimum cost in the static condensation. Moreover, τ_{it} has a significant contribution to the total time cost. This contribution is growing as the macro-elements sizes decrease. Ultimately, the reduction of the computational time obtained by using the hybrid solver is approximately a factor of two, independently of the polynomial degree.

When the problem size increases to 4096^2, both, the number of FLOPs and the total computational time show a similar behavior to the problem with a mesh size of 2048^2. In this case, the reduction factor of the total computational time increases and becomes slightly above three.

This is a result of the decrease of the iterative solver computational time (Tables 5.14 and 5.15). For sufficiently large problems, the reduction factor associated to the skeleton system solution (iterative step) will determine the total reduction factor of rIGA with respect to IGA, since that is the only term scaling non-linearly with respect to the problem size.

n_{elem}	p	s	θ_{SC}	θ_{pre}	θ_{it}	θ
4096	2	4096 (IGA)	***	5.05e+09	7.04e+11	7.09e+11
		128	1.15e+12	1.03e+11	4.08e+10	1.29e+12
		64	5.58e+11	5.43e+10	6.76e+10	6.80e+11
		32	2.80e+11	2.89e+10	1.11e+11	4.19e+11
		16	1.24e+11	1.60e+10	1.62e+11	3.02e+11
	3	4096 (IGA)	***	1.97e+10	1.30e+12	1.32e+12
		128	3.00e+12	1.06e+11	3.85e+10	3.15e+12
		64	1.53e+12	5.68e+10	5.76e+10	1.65e+12
		32	7.86e+11	3.16e+10	1.04e+11	9.22e+11
		16	3.95e+11	1.90e+10	1.95e+11	6.09e+11

Table 5.14.: Number of FLOPs required to perform static condensation (θ_{SC}), construct the preconditioner (θ_{pre}), compute the skeleton system solution (θ_{it}) and solve the 2D Poisson model problem (θ).

n_{elem}	p	s	τ_{SC}	τ_{pre}	τ_{it}	τ
4096	2	4096 (IGA)	***	9.77	492.65	502.42
		128	154.79	45.58	26.20	226.55
		64	108.09	25.74	43.42	177.25
		32	78.55	15.45	70.84	164.84
		16	73.67	11.13	103.46	188.26
	3	4096 (IGA)	***	23.93	780.50	804.43
		128	297.31	46.59	24.69	368.60
		64	196.75	26.82	37.06	260.63
		32	137.44	17.97	65.92	221.32
		16	104.70	12.86	123.77	241.32

Table 5.15.: Computational time (in seconds) spent to perform static condensation (τ_{SC}), construct the preconditioner (τ_{pre}), compute the skeleton system solution (τ_{it}) and solve the 2D Poisson model problem (τ).

Having a reduction factor greater in computational time than in the number of FLOPs can be an effect of the domain partitioning. The partitioning improves the performance of the transfer of data. This benefits the cache and results in a speed-up of the computational time. Large sparse systems degrade the access to memory and transfer data performance. In those cases, the machine has more probabilities to fail in optimally filling the cache. This results in delay

operations until the required data is transferred. The division of the problem in sub-problems (macro-elements) permits to improve the performance of the cache-based machines. In particular, the solution of all the small problems results in a smaller global system that better exploit the data locality, and benefits the filling of the cache [51].

In the hybrid solver strategy, the cost of static condensation of the macro-elements (that corresponds to compute the Schur complement of IGA systems) depends on the number of internal macro-elements DoF and the polynomial degree. Moreover, the preconditioner construction, that consists of an incomplete factorization of all DoF at the macro-elements boundaries (DoF of the skeleton system), involves a cost that depends on the NZ entries of the skeleton matrix system. Likewise, the matrix-vector multiplications performed per iteration of the iterative solver (used to compute the solution of the skeleton system) have a cost proportional to the number of NZ entries of the skeleton matrix system. Therefore, the cost of those operations (static condensation, preconditioner construction, and matrix vector product) relies only on the problem discretization (mesh size, polynomial order, and continuity degree) and provide a similar performance independently of the problem we are solving.

In well-conditioned problems (matrices with small condition numbers), the cost of static condensation governs the total cost, in particular for high polynomial degrees, and slight reductions in the total cost are obtained, as observed in the numerical results. The hybrid solver strategy delivers larger computational savings with respect to C^{p-1} IGA when the problem is discretized with huge mesh sizes or when solving more complex problems, e.g., Helmholtz. In those cases, the performance of traditional iterative solvers degrades as the condition number of the system matrix enlarges, resulting in slow convergence velocities. However, once we implement the hybrid solver strategy, we obtain a skeleton (reduced) system that is not only smaller than the original matrix, but it also involves a matrix with a lower number of NZ entries and a better conditioning than the original one. Considering that the costs of static condensation, preconditioning construction and matrix-vector product are independent of the problem, the hybrid solver strategy will deliver larger computational savings than C^{p-1} IGA.

5.3.5. 3D numerical results

The extension of the rIGA hybrid solver to 3D is straightforward, since no significant modifications in the implementation presented in section 4.3 are required to build the solver. Unfortunately, in 3D, the number of NZ entries of matrix A_{skl} increases with respect to those of A (see Table 5.17), as opposed to what we encounter in the 2D case (see Tables 5.8 and 5.16).

p	s	Matrix size (DoF)			Number of NZ entries		
		A	A_{skl}	A/A_{skl}	A	A_{skl}	A/A_{skl}
2	2048 (IGA)	4.20E+06	–	–	1.05E+08	–	–
	512	–	2.05E+04	204.95	–	6.10E+07	1.72
	128	–	6.99E+04	60.10	–	6.00E+07	1.75
	32	–	2.70E+05	15.54	–	6.21E+07	1.69
	8	–	1.12E+06	3.76	–	7.19E+07	1.46
	2	–	5.25E+06	0.80	–	1.19E+08	0.88
6	2048 (IGA)	4.22E+06	–	–	7.13E+08	–	–
	512	–	2.07E+04	204.16	–	6.20E+07	11.50
	128	–	7.21E+04	58.52	–	6.38E+07	11.17
	32	–	3.04E+05	13.89	–	7.81E+07	9.13
	8	–	1.65E+06	2.56	–	1.52E+08	4.70
	2	–	1.36E+07	0.31	–	6.89E+08	1.03

Table 5.16.: Comparison between the number of NZ entries of the original matrix A and that of the skeleton matrix A_{skl} in 2D. A mesh size of $N_{\text{elem}} = 2048^2$ is assumed.

p	s	Matrix size (DoF)			Number of NZ entries		
		A	A_{skl}	A/A_{skl}	A	A_{skl}	A/A_{skl}
2	2048 (IGA)	8.62E+09	–	–	1.08E+12	–	–
	512	–	6.31E+07	136.600	–	1.50E+14	0.00720
	128	–	2.16E+08	39.942	–	3.75E+13	0.02868
	32	–	8.44E+08	10.206	–	1.02E+13	0.10594
	8	–	3.66E+09	2.356	–	3.50E+12	0.30777
	2	–	2.04E+10	0.422	–	2.59E+12	0.41518
6	2048 (IGA)	8.67E+09	–	–	1.90E+13	–	–
	512	–	6.41E+07	135.28	–	1.54E+14	0.12
	128	–	2.29E+08	37.79	–	4.24E+13	0.45
	32	–	1.06E+09	8.14	–	1.61E+13	1.18
	8	–	7.90E+09	1.10	–	1.54E+13	1.24
	2	–	1.37E+11	0.06	–	8.11E+13	0.23

Table 5.17.: Comparison between the number of NZ entries of the original matrix A and that of the skeleton matrix A_{skl} in 3D. A mesh size of $N_{\text{elem}} = 2048^3$ is assumed.

In 3D, the number of NZ entries per column of matrix A_{skl} is approximately $11(s+p)^2$. This value is larger than the number of NZ entries per column of matrix A, that is $(2p+1)^3$. Moreover, for $s > p$ (the most important scenario), the number of NZ entries becomes gigantic. For instance, in a problem discretized with 2048^3 elements and a polynomial degree $p = 2$, using macro-elements of size 32^3, we obtain in average 12085 NZ entries per column (unknown). This is approximately two orders of magnitude larger than the 125 NZ entries per column observed when using traditional IGA.

Large numbers of NZ entries imply that the cost of a single CG iteration to approximate the solution of the skeleton system is more expensive than a single CG iteration to approximate the solution of the full problem discretized with C^{p-1} IGA without static condensation. Therefore, the hybrid solver strategy combined with rIGA is unsuitable to solve 3D problems. An alternative approach is needed.

6. Numerical Applications

In this chapter, we apply refined Isogeometric Analysis (rIGA) to simulate the flow of an incompressible fluid on bounded domains. This is a multi-field problem since the model includes the pressure and the vectorial velocity of the fluid. We describe the Isogeometric Analysis (IGA) and rIGA discretizations used to solve the fluid flow problem, and we discuss how to perform the continuity reduction to solve the multi-fields problem. Then, we provide an estimate of the FLoating Point Operations (FLOPs) required to solve the incompressible fluid flow problem with those discretizations. Lastly, we detail the model problem implementation, and we present some numerical results.

6.1. Fluid flow model problem

The model problem consists of a linear system of Partial Differential Equations (PDEs) for the conservation of linear momentum and mass. We assume that this boundary value problem is steady state. Equation 6.1 provides the system used to model the incompresible fluid flow.

$$\begin{cases} \text{Find } \{\mathbf{u}, p_e\}\,, \text{ with } \mathbf{u} : \Omega \to \mathbb{R}^d, \text{ and } p_e : \Omega \to \mathbb{R}, \text{ such that:} \\ \qquad\qquad \beta_{rec}\mathbf{u} - \nabla \cdot \sigma\left(\mathbf{u}, p_e\right) = \mathbf{f} \qquad \text{in } \Omega, \\ \qquad\qquad \nabla \cdot \mathbf{u} = \mathbf{0} \qquad \text{in } \Omega, \\ \qquad\qquad \mathbf{u} = \mathbf{g} \qquad \text{on } \partial\Omega, \end{cases} \tag{6.1}$$

where $\Omega = (0,1)^d$ is the region occuped by the fluid, $\mathbf{u}$ is the fluid velocity field, p_e is the pressure field, $\mathbf{f}$ is the external volumetric forces acting over the fluid, and $\mathbf{g}$ is the Dirichlet boundary condition that can be decomposed into the normal and tangential components, $\mathbf{g} = \mathbf{g}_n + \mathbf{g}_t$. Additionally, $\sigma\left(\mathbf{u}, p_e\right) = -p_e\mathbf{I} + 2\nu\nabla^s\mathbf{u}$ is the Cauchy stress tensor for incompressible fluids, where $\mathbf{I}$ refers to the identity matrix, ν is the kinematic viscosity, and $\nabla^s\mathbf{u}$ is the symmetric part of the velocity gradient (strain rate). β_{rec} is a reaction coefficient that denotes the ratio of the fluid viscocity to the fluid permeability. Table 6.1 shows the different incompressible fluid flow problems we can solve using Equation 6.1.

The weak formulation of the system is

$$\begin{cases} \text{Find } \{\mathbf{u}, p_e\}\,, \text{ with } \mathbf{u} \in V_g, \text{ and } p_e \in Q_0, \text{ such that:} \\ \qquad (\nabla^s\mathbf{w}, 2\nu\nabla^s\mathbf{u})_\Omega + (\mathbf{w}, \beta_{rec}\mathbf{u})_\Omega - (\nabla\cdot\mathbf{w}, p_e)_\Omega + (q_e, \nabla\cdot\mathbf{u})_\Omega = (\mathbf{w}, \mathbf{f})_\Omega \\ \text{for all } \mathbf{w} \in V_0, \text{ and } q_e \in Q_0, \end{cases} \tag{6.2}$$

where $V_g = \{\mathbf{v} \in \mathbf{H}^1(\Omega) : \mathbf{v} = \mathbf{g} \text{ on } \partial\Omega\}$ and $V_0 = \{\mathbf{v} \in \mathbf{H}^1(\Omega) : \mathbf{v} = 0 \text{ on } \partial\Omega\}$ are the trial and test spaces for the velocity field, respectively. The trial and test space for the pressure field is $Q_0 = L^2_0(\Omega) \subset L^2(\Omega)$ that corresponds to a space with zero average on Ω. In here, $(\cdot\cdot)_\Omega$ denotes the L^2 inner product on Ω. The weak problem has a unique solution $(\mathbf{u}, p_e) \in V_g \times Q_0$.

Fluid flow problem	Diffusion-reaction ($\beta_{rec} \boxdot \nu$)
Darcy problem	$\beta_{rec} \gg \nu$
Brinkman problem	$\beta_{rec} \simeq \nu$
Stokes problem	$\beta_{rec} \ll \nu$

Table 6.1.: Incompressible fluid flow problems solved with Equation 6.1.

6.2. IGA discretization

In order to solve the problem in Equation 6.1 with IGA, we build the problem discretization of the weak formulation (Equation 6.2) based on the discretization framework presented in previous works [29, 30, 110, 58, 59, 60]. In particular, we use a spline generalization of the Raviart-Thomas Finite Element Analysis (FEA) spaces to approximate the velocity field. These are globally continuous discrete spaces due to the high inter-element smoothness that the splines provide. Therefore, the discretization framework is conforming. Moreover, the smooth Raviart-Thomas spaces satisfy the inf-sup stability condition and guarantee divergence-free discrete solutions [29]. The spaces for the velocity and the pressure in the parametric domain ($\hat{\Omega}$) are as follows:

$$\text{Velocity spaces: } \hat{\mathcal{V}}_h(\hat{\Omega}) = \begin{cases} \mathcal{S}^{p+1,p}_{k+1,k} \times \mathcal{S}^{p,p+1}_{k,k+1} & \text{Surface (2D)} \\ \mathcal{S}^{p+1,p,p}_{k+1,k,k} \times \mathcal{S}^{p,p+1,p}_{k,k+1,k} \times \mathcal{S}^{p,p,p+1}_{k,k,k+1} & \text{Solid (3D)} \end{cases} \tag{6.3}$$

$$\text{Pressure space: } \hat{\mathcal{Q}}_h(\hat{\Omega}) = \begin{cases} \mathcal{S}^{p,p}_{k,k} & \text{Surface (2D)} \\ \mathcal{S}^{p,p,p}_{k,k,k} & \text{Solid (3D)} \end{cases} \tag{6.4}$$

In Figure 6.1, we illustrate the smooth Raviart-Thomas spaces for the velocity and the pressure fields using Non-uniform Rational Basis Splines (NURBS) of degree $p = 2$ and continuity $k = 1$ associated to a 2D mesh of size of 6×6 elements. Considering that we focus on using highly continuous discretizations (C^{p-1} IGA) with local continuity reduction but keeping inter-element regularity ($C^k : k \geqslant 0$), then the discrete velocity field is $\mathbf{H}^1$-conforming, being $\mathbf{H}^1 = \left(H^1\right)^{d_f}$ and d_f is the velocity field dimension. Moreover, since the univariate B-splines have the stronger relationship

$$\mathcal{C}^p_k \equiv \left\{ \frac{\partial}{\partial \xi} v : v \in \mathcal{C}^{p+1}_{k+1} \right\},$$

then for any $p \geqslant 1$ and $0 \geqslant k \geqslant p - 1$

$$\left\{ \nabla \cdot \hat{\mathbf{v}} : \hat{\mathbf{v}} \in \hat{\mathcal{V}}_h(\hat{\Omega}) \right\} = \hat{\mathcal{Q}}_h(\hat{\Omega}). \tag{6.5}$$

Thus, the discretization using smooth Raviart-Thomas spaces provides a discrete divergence-free velocity field.

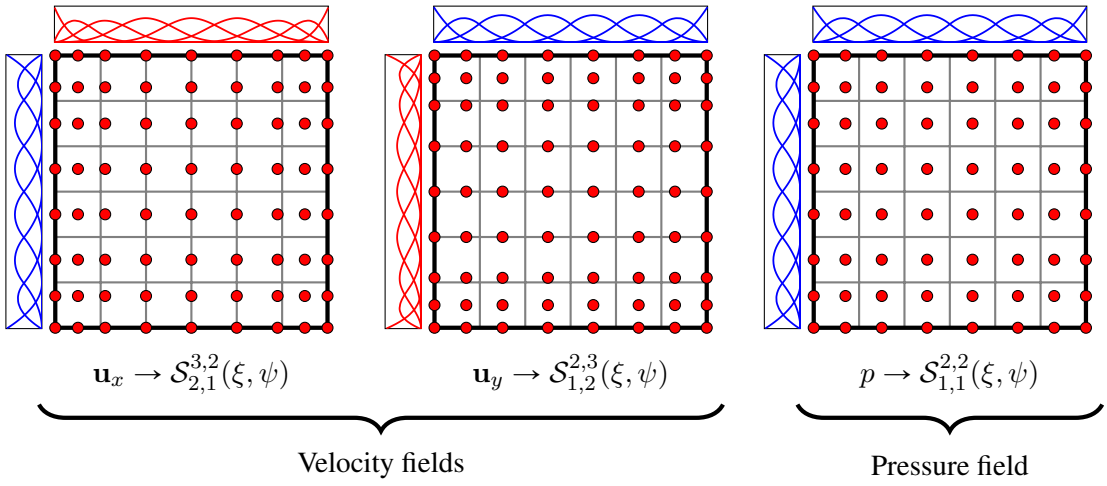


Figure 6.1.: Example of the smooth Raviart-Thomas spaces for the velocity and the pressure fields. In this case, we show the spaces for a 2D discretization with a mesh size of 6×6 elements, polynomial order $p = 2$ and continuity degree $k = 1$.

We use the following mappings to define the discrete velocity and pressure spaces on the physical domain

$$\begin{aligned} \iota_{\mathbf{u}} &= det(D\mathbf{F})(D\mathbf{F})^{-1}(\mathbf{v} \circ \mathbf{F}) \qquad && \mathbf{v} \in \mathbf{H}(div;\Omega) \cap \mathbf{H}^1(\Omega), \\ \iota_{p_e} &= det(D\mathbf{F})(q_e \circ \mathbf{F}) && q_e \in L^2_0(\Omega), \end{aligned}$$

where *det* and *div* refer to the determinant and the divergence, respectively. $\iota_{\mathbf{u}}$ is a divergence-preserving map known as Piola transformation, and ι_p is an integral-preserving transformation. In here, $\mathbf{F}$ is the geometric mapping from the parametric domain to the physical domain, and $D\mathbf{F}$ is the gradient of the mapping. Therefore, the discrete spaces in the physical domain are denoted as

$$\mathcal{V}_h := \left\{ \mathbf{v} \in \mathbf{H}(div;\Omega) \cap \mathbf{H}^1(\Omega) : \iota_{\mathbf{u}}(\mathbf{v}) \in \hat{\mathcal{V}}_h(\hat{\Omega}) \right\},$$

$$\mathcal{Q}_h := \left\{ q_e \in L^2_0(\Omega) : \iota_{p_e}(q_e) \in \hat{\mathcal{Q}}_h(\hat{\Omega}) \right\}.$$

These discrete spaces inmediately satisfy the inf-sup condition, that is

$$\inf_{q_e \in \mathcal{Q}_h, q_e \neq 0} \sup_{\mathbf{v} \in \mathcal{V}_h} \frac{(\nabla \cdot \mathbf{v}, q_e)_\Omega}{||\mathbf{v}||_{\mathbf{H}^1} \, ||q_e||_{L^2}} \geqslant C_{is} > 0,$$

where C_{is} is a constant independent of the element size.

6.2.1. Boundary condition imposition

Since the discretization spaces used to perform the discretization are designed for $\mathbf{H}(div)$ problems (Raviart-Thomas [107]), we can easily set the normal component of the velocity to zero

($\mathbf{u}_n$) at the boundaries, by removing the corresponding Degrees of Freedom (DoF). The parametric spaces, in this case, read as

$$\hat{\mathcal{V}}_{0,h} := \left\{ \hat{\mathbf{v}} \in \hat{\mathcal{V}}_h(\hat{\Omega}) : \hat{\mathbf{v}} \cdot \hat{\mathbf{n}} = 0 \text{ on } \partial\hat{\Omega} \right\},$$

$$\hat{\mathcal{Q}}_{0,h} := \left\{ \hat{q}_e \in \hat{\mathcal{Q}}_h(\hat{\Omega}) : \int_\Omega \hat{q}_e = 0 \right\}.$$

It is hard to strongly impose the tangential component of the velocity ($\mathbf{u}_t$). Therefore, we use Nitsche's method to weakely set the tangential component of the Dirichlet boundary condition [98]. In Equation 6.6, we introduce the Galerkin formulation that use the weak imposition of the tangential component of the velocity.

$$\begin{cases} \text{Find } \{\mathbf{u}, p_e\}, \text{ with } \mathbf{u} \in \mathcal{V}_{0,h}, \text{ and } p_e \in \mathcal{Q}_{0,h}, \text{ such that:} \\ \quad (\nabla^s \mathbf{w}, 2\nu\nabla^s \mathbf{u})_\Omega + (\mathbf{w}, \beta_{rec}\mathbf{u})_\Omega \\ \qquad -(\nabla\cdot\mathbf{w}, p_e)_\Omega + (q_e, \nabla\cdot\mathbf{u})_\Omega \qquad\qquad (\mathbf{w}, \mathbf{f})_\Omega \\ \qquad\qquad -(\mathbf{w}, 2\nu\nabla^s\mathbf{u}\cdot\mathbf{n})_\Gamma \quad = \\ \qquad\qquad +(\mathbf{w}, 2\nu\mathbf{u}\,\alpha_p)_\Gamma \qquad\quad +(\mathbf{w}, 2\nu\mathbf{g}\,\alpha_p)_\Gamma \\ \qquad\qquad -(\mathbf{u}, 2\nu\nabla^s\mathbf{w}\cdot\mathbf{n})_\Gamma \qquad\quad -(\mathbf{g}, 2\nu\nabla^s\mathbf{w}\cdot\mathbf{n})_\Gamma \\ \text{for all } \mathbf{w} \in \mathcal{V}_{0,h}, \text{ and } q_e \in \mathcal{Q}_{0,h}, \end{cases} \tag{6.6}$$

where Γ refers to the boundary in which the tangential condition is impossed, $\mathbf{g}$ is the boundary condition that consists in a tangential value $\mathbf{g}_t$ and a zero normal value $\mathbf{g}_n = \mathbf{0}$, and $\alpha_p = C_{pen}/h_f$. $C_{pen} = 5(p+1)$ is the penalty parameter that depends of the polynomial degree of the discretization, and h_f is the wall normal mesh size [18]. In the Galerkin formulation, the term in blue comes from the natural boundary condition (consistency), the penalization terms are in red and the terms in green conform the adjoint consistency.

6.2.2. Computational complexity for direct solvers

We use the state-of-the-art multifrontal method to solve the weak formulation of the multi-field problem. As we explain in section 3.1.1, the multifrontal solver partitions the domain into macro-elements interconnected by the separators. The size of the macro-elements and separators in a multi-field case is equal to the sum of unknowns that each of those subdomains contains in each field. As an example, we show in Figure 6.2 the size of the separator at the first partition level (vertical partition) used to split a 2D domain into two subdomains.

The cost of the LU factorization is dominanted by the cost of eliminating the DoF on the separators [41, 38].

6.2.2.1. Bi-dimensional case

In 2D, the size of the separator (vertical separator) at the first partition (vertical cut) is given by

$$\begin{aligned} q_{sep}|_y &= q_{sep}^{\mathbf{u}_x}|_y + q_{sep}^{\mathbf{u}_y}|_y + q_{sep}^{p_e}|_y \\ &= \mathcal{O}\left(n_{\mathbf{u}_x}|_y(p+1) + n_{\mathbf{u}_y}|_y(p) + n_{p_e}|_y(p) \right), \end{aligned} \tag{6.7}$$

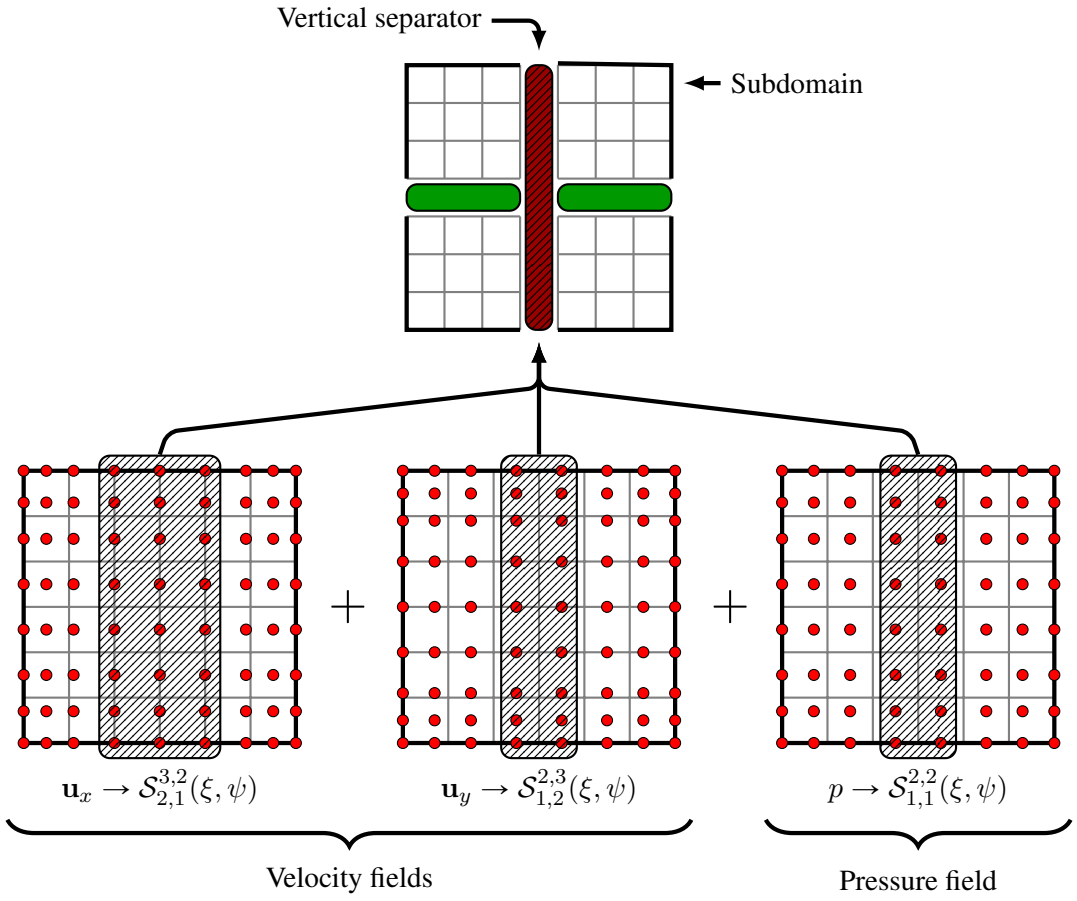


Figure 6.2.: Illustration of the size of the vertical separator that splits a mesh of 6×6 elements into two symmetric subdomains. The NURBS spaces have a polynomial order $p = 2$ and a continuity degree $k = 1$.

where $n_{\mathbf{u}_x}|_y = (n_{\text{elem}} + p)$, $n_{\mathbf{u}_y}|_y = (n_{\text{elem}} + p + 1)$ and $n_{p_e}|_y = (n_{\text{elem}} + p)$ are the length of the separator in each space. Then, replacing the above equalities into Equation 6.7, we obtain

$$\begin{aligned} q_{sep}|_y &= \mathcal{O}\left(n_{\mathbf{u}_x}|_y(p+1) + n_{\mathbf{u}_y}|_y(p) + n_{p_e}|_y(p)\right) \\ &= \mathcal{O}\left((n_{\text{elem}} + p)(p+1) + (n_{\text{elem}} + p + 1)(p) + (n_{\text{elem}} + p)(p)\right) \\ &= \mathcal{O}\left((n_{\text{elem}} + p)(3p+1) + p\right). \end{aligned} \tag{6.8}$$

The size of both (horizontal) separators at the second partition (horizontal cut) is given by

$$\begin{aligned} q_{sep}|_x &= q^{\mathbf{u}_x}_{sep}|_x + q^{\mathbf{u}_y}_{sep}|_x + q^{p_e}_{sep}|_x \\ &= \mathcal{O}\left(n_{\mathbf{u}_x}|_x(p) + n_{\mathbf{u}_y}|_x(p+1) + n_{p_e}|_x(p)\right). \end{aligned} \tag{6.9}$$

The length of these separators in each space is $n_{\mathbf{u}_x}|_x \approx 0.5(n_{\text{elem}} + p + 1)$, $n_{\mathbf{u}_x}|_y \approx$

$0.5(n_{\text{elem}} + p)$ and $n_{p_e}|_x \approx 0.5(n_{\text{elem}} + p)$, respectively. Therefore,

$$\begin{aligned} q_{sep}|_x &= \mathcal{O}\left(n_{\mathbf{u}_x}|_x(p) + n_{\mathbf{u}_y}|_x(p+1) + n_{p_e}|_x(p)\right) \\ &= \mathcal{O}\left(0.5(n_{\text{elem}} + p + 1)(p) + 0.5(n_{\text{elem}} + p)(p+1) + 0.5(n_{\text{elem}} + p)(p)\right) \\ &= \mathcal{O}\left(0.5(n_{\text{elem}} + p)(3p+1) + p\right). \end{aligned} \tag{6.10}$$

The size of the separators in *all* partition levels can be computed based on the previous mathematical expressions (Equations 6.8 and 6.10) by following the recursive partition of the domain. Then, the size of the separators at the i-th partition level is

$$q_{sep}|_y = \mathcal{O}\left(2^{-(i-1)}\left((n_{\text{elem}} + p)(3p+1) + p\right)\right), \tag{6.11}$$

$$q_{sep}|_x = \mathcal{O}\left(2^{-i}\left((n_{\text{elem}} + p)(3p+1) + p\right)\right). \tag{6.12}$$

Finally, the cost to solve the LU factorization in 2D is given by

$$\begin{aligned} \theta_{\text{IGA}}|_{\text{2D}} &= \sum_{i=1}^{\ell} 2^{2(i-1)}\left(2^{-(i-1)}\left((n_{\text{elem}} + p)(3p+1) + p\right)\right)^3 \\ &\qquad + 2^{2(i-1)+1}\left(2^{-(i)}\left((n_{\text{elem}} + p)(3p+1) + p\right)\right)^3, \\ \theta_{\text{IGA}}|_{\text{2D}} &= \frac{5}{2}\left(1 - 2^{-\ell}\right)\left((n_{\text{elem}} + p)(3p+1) + p\right)^3, \\ \theta_{\text{IGA}}|_{\text{2D}} &= \mathcal{O}\left(\left((n_{\text{elem}} + p)(3p+1) + p\right)^3\right). \end{aligned} \tag{6.13}$$

6.2.2.2. d-dimensional case

Following the same procedure that we use to estimate the cost of LU factorization in 2D, we build the estimate for 3D, that is

$$\theta_{\text{IGA}}|_{\text{3D}} = \mathcal{O}\left(\left((n_{\text{elem}} + p)^2(4p+1) + 2(n_{\text{elem}} + p)p\right)^3\right). \tag{6.14}$$

Based on the cost estimates for 2D and 3D, we build an expresion that computes the estimate of the LU factorization cost for a problem of dimension $d > 1$. The expression is

$$\theta_{\text{IGA}} = \mathcal{O}\left(\left((n_{\text{elem}} + p)^{(d-1)}((d+1)p+1) + (d-1)(n_{\text{elem}} + p)^{(d-2)}p\right)^3\right). \tag{6.15}$$

6.3. rIGA discretization

rIGA reduces the continuity over specific inter-element boundaries that correspond with the location of the separators at different partition levels. To guarantee that the resulting rIGA discretization preserves the discrete divergence-free velocity field $(\nabla \cdot \mathbf{u}, q_e)_\Omega = 0$, we reduce

the continuity over the element interfaces in k degrees on all spaces. Therefore, in a rIGA discretization, the discrete spaces are

$$\hat{\mathcal{V}}_h := \left\{ \mathcal{S}^{p+1,p}_{k+1-k|_{v_s},k-k|_{h_s}} \times \mathcal{S}^{p,p+1}_{k-k|_{v_s},k+1-k|_{h_s}} \right\},$$

$$\hat{\mathcal{Q}}_h := \left\{ \mathcal{S}^{p,p}_{k-k|_{v_s},k-k|_{h_s}} \right\},$$

where v_s and h_s refers to the vertical and horizontal separators that involve continuity reduction, respectively. For instance, in the space of u_x, that is

$$\mathcal{S}^{p+1,p}_{k+1-k|_{v_s},k-k|_{h_s}},$$

rIGA employs C^1-hyperplanes along the vertical macro-element interfaces, reducing the continuity degree to one, and uses C^0-hyperplanes along the horizontal macro-element interfaces, reducing the continuity until reaching zero degree.

Figure 6.3 illustrates the spaces of the velocity and pressure fields once we reduce the continuity at the first vertical and horizonal cuts that split a 2D mesh of 6×6 elements into four subdomains of 3×3 elements.

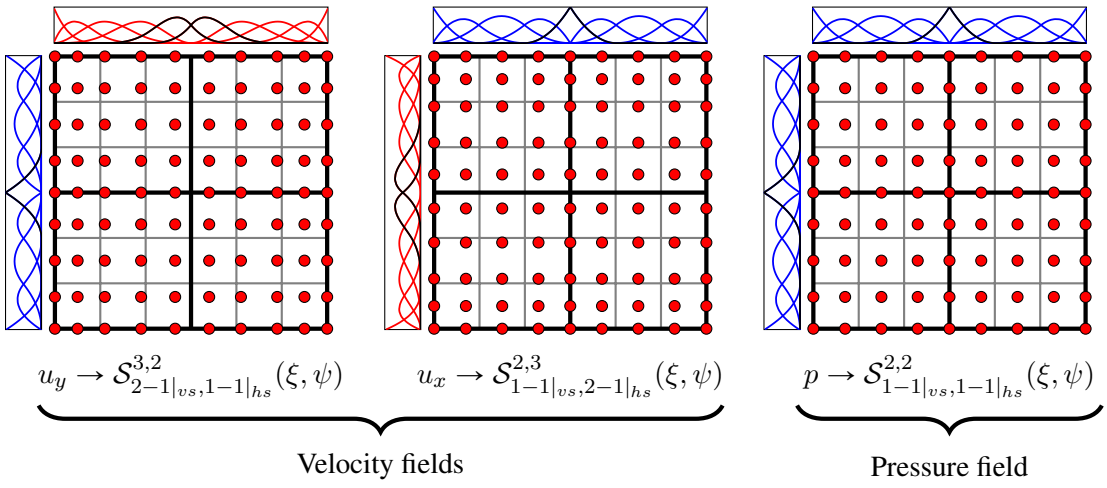


Figure 6.3.: Velocity and pressure spaces after partitioning a 2D mesh composed of 6×6 elements into four macro-elements of 3×3 elements. The polynomial degree is $p = 2$. v_s and h_s refer to the vertical and horizontal separators, respectively. The partition level reduces the continuity of the velocity and pressure spaces by one degree along the inter-subdomains boundaries.

6.3.1. Computational complexity for direct solvers

In the case of rIGA, the computational cost of the LU factorization is equal to the cost of eliminating the DoF on the separators and macro-elements [67, 65, 66].

6.3.1.1. Bidimensional case

The size of a separator is equal to the sum of the unknowns contained in each field. The reduction of continuity modifies this size, reducing the thickness and increasing separators' length. The size of the separator at the first partition (vertical cut) is given by

$$q_{sep}|_y = q_{sep}^{\mathrm{u}_x}|_y + q_{sep}^{\mathrm{u}_y}|_y + q_{sep}^{p_e}|_y. \tag{6.16}$$

This separator corresponds to a C^1-hyperplane in u_x, while in u_y and p_e, it corresponds to a C^0-hyperplane. Thus,

$$q_{sep}|_y = \mathcal{O}\left(n_{\mathrm{u}_x}|_y(2) + n_{\mathrm{u}_y}|_y(1) + n_{p_e}|_y(1)\right). \tag{6.17}$$

The length of the separator in each space is $n_{\mathrm{u}_x}|_y = (n_{\text{elem}} + p + (2^i - 1)(p-1))$, $n_{\mathrm{u}_y}|_y = (n_{\text{elem}} + p + 1 + (2^i - 1)(p-1))$ and $n_{p_e}|_y = (n_{\text{elem}} + p + (2^i - 1)(p-1))$, respectively. The additional term (last term inside the parenthesis) in $n_{\mathrm{u}_x}|_y$, $n_{\mathrm{u}_y}|_y$ and $n_{p_e}|_y$ corresponds to the number of unknowns added to the separators length due to the reduction of the continuity at i partition levels. The size of the vertical separator is then

$$q_{sep}|_y = \mathcal{O}\left((n_{\text{elem}} + p + (2^i - 1)(p-1))(4) + (1)\right). \tag{6.18}$$

The size of the separators at the second partition (horizontal cut) is given by

$$q_{sep}|_x = q_{sep}^{\mathrm{u}_x}|_x + q_{sep}^{\mathrm{u}_y}|_x + q_{sep}^{p_e}|_x. \tag{6.19}$$

In this case, the separators correspond to a C^1-hyperplane in u_y. In the remaining spaces, the separatos correspond to a C^0-hyperplane. Therefore,

$$q_{sep}|_x = \mathcal{O}\left(n_{\mathrm{u}_x}|_x(1) + n_{\mathrm{u}_y}|_x(2) + n_{p_e}|_x(1)\right), \tag{6.20}$$

where the separators length is

$$\begin{aligned} n_{\mathrm{u}_x}|_x &\approx 0.5(n_{\text{elem}} + p + 1 + (2^i - 1)(p-1)) \\ n_{\mathrm{u}_y}|_x &\approx 0.5((n_{\text{elem}} + p + (2^i - 1)(p-1)) \\ n_{p_e}|_x &\approx 0.5(n_{\text{elem}} + p + (2^i - 1)(p-1)). \end{aligned}$$

Then, the separators size is given by

$$q_{sep}|_x = \mathcal{O}\left(0.5(n_{\text{elem}} + p + (2^i - 1)(p-1))(4) + 0.5\,(1)\right). \tag{6.21}$$

We derive a general formula to compute the size of the separators at *all* partition levels based on Equations 6.18 and 6.21. Then, the size of the separators at the i-th partition level is

$$q_{sep}|_y = \mathcal{O}\left(2^{-(i-1)}\left((n_{\text{elem}} + p + (2^i - 1)(p-1))(4) + 1\right)\right), \tag{6.22}$$

$$q_{sep}|_x = \mathcal{O}\left(2^{-(i)}\left((n_{\text{elem}} + p + (2^i - 1)(p-1))(4) + 1\right)\right), \tag{6.23}$$

and the cost to eliminate the DoF on the separators is

$$\begin{aligned}\theta_{sep}|_{\text{2D}} = \sum_{i=1}^{\ell} 2^{2(i-1)} \Big(2^{-(i-1)}\Big((n_{\text{elem}}+p+(2^i-1)(p-1))(4)+1\Big)\Big)^3 \\ + 2^{2(i-1)+1}\Big(2^{-i}\left((n_{\text{elem}}+p+(2^i-1)(p-1))(4)+1\right)\Big)^3, \\ \theta_{sep}|_{\text{2D}} = \frac{5}{2}\Big(1-2^{-\ell}\Big)\Big((n_{\text{elem}}+p+(2^\ell-1)(p-1))(4)+1\Big)^3, \\ \theta_{sep}|_{\text{2D}} = \mathcal{O}\left(\left((n_{\text{elem}}+p+(2^\ell-1)(p-1))(4)+1\right)^3\right).\end{aligned} \tag{6.24}$$

where ℓ is the number of partition levels that involve continuity reduction.

Next, we estimate the cost to eliminate the DoF on the macro-elements. We assume that the macro-elements are C^{p-1} systems. The size of the macro-elements is $n_{\text{m-e}} = n_{\text{m-e}}|_x \cdot n_{\text{m-e}}|_y$, being $n_{\text{m-e}}|_x$ and $n_{\text{m-e}}|_y$ the size of the macro-element in the horizontal and vertical spatial directions, respectively. Table 6.2 provides the macro-elements sizes for the velocity and pressure fields.

	$n_{\text{m-e}}\|_x$	$n_{\text{m-e}}\|_y$
u_x	$2^{-\ell}\left(n_{\text{elem}}+(p+1)+(2^\ell-1)(p-1)\right)$	$2^{-\ell}\left(n_{\text{elem}}+p+(2^\ell-1)(p-1)\right)$
u_y	$2^{-\ell}\left(n_{\text{elem}}+p+(2^\ell-1)(p-1)\right)$	$2^{-\ell}\left(n_{\text{elem}}+(p+1)+(2^\ell-1)(p-1)\right)$
p_e	$2^{-\ell}\left(n_{\text{elem}}+p+(2^\ell-1)(p-1)\right)$	$2^{-\ell}\left(n_{\text{elem}}+p+(2^\ell-1)(p-1)\right)$

Table 6.2.: Macro-element size in the velocity and pressure fields.

The cost to eliminate the DoF in a single macro-element is given by an adaptation of Equation 6.15 that includes the unknowns added to the discretization due to the reduction of the continuity at the ℓ partition levels, and reads as

$$\theta_{\text{m-e}}|_{\text{2D}} = 2^{-3\ell}\mathcal{O}\left(\left((n_{\text{elem}}+p+(2^\ell-1)(p-1))(3p+1)+p\right)^3\right). \tag{6.25}$$

The cost of factorizing *all* the macro-elements is $2^{2\ell}\cdot\theta_{\text{m-e}}$, being $2^{2\ell}$ the number of macro-elements. Finally, the total cost of the LU factorization when using rIGA to build the discretization of the velocity and pressure fields is given by

$$\begin{aligned}\theta_{\text{rIGA}}|_{\text{2D}} = \theta_{sep}|_{\text{2D}} + 2^{2\ell}\cdot\theta_{\text{m-e}}|_{\text{2D}}, \\ \theta_{\text{rIGA}}|_{\text{2D}} = 2^{-\ell}\mathcal{O}\left(\left((n_{\text{elem}}+p+(2^\ell-1)(p-1))(3p+1)+p\right)^3\right) \\ + \mathcal{O}\left(\left((n_{\text{elem}}+p+(2^\ell-1)(p-1))(4)+1\right)^3\right).\end{aligned} \tag{6.26}$$

6.3.1.2. d-dimensional case

We follow the same procedure that we use to estimate the cost of the LU factorization when using rIGA in 2D in order to build a formula that estimates the LU factorizaton cost in 3D. The resulting formula is

$$\begin{aligned}
\theta_{\text{rIGA}}|_{\text{3D}} &= \theta_{sep}|_{\text{3D}} + 2^{3\ell} \cdot \theta_{\text{m-e}}|_{\text{3D}}, \\
\theta_{\text{rIGA}}|_{\text{3D}} =& 2^{-3\ell}\mathcal{O}\left(\left((n_{\text{elem}}+p+(2^{\ell}-1)(p-1))^2(4p+1)\right.\right.\\
&\qquad\left.\left.+2(n_{\text{elem}}+p+(2^{\ell}-1)(p-1))p\right)^3\right)\\
&+\mathcal{O}\left(\left((n_{\text{elem}}+p+(2^{\ell}-1)(p-1))^2(5)\right.\right.\\
&\qquad\left.\left.+2(n_{\text{elem}}+p+(2^{\ell}-1)(p-1))\right)^3\right).
\end{aligned} \tag{6.27}$$

Based on the estimates for 2D and 3D, we derive a mathematical expresion to approximate the LU factorization cost for problems of dimension $d > 1$, and that is

$$\begin{aligned}
\theta_{\text{rIGA}} &= \theta_{sep} + 2^{d\ell} \cdot \theta_{\text{m-e}}, \\
\theta_{\text{rIGA}} =& 2^{(3-2d)\ell}\mathcal{O}\left(\left((n_{\text{elem}}+p+(2^{\ell}-1)(p-1))^{(d-1)}((d+1)p+1)\right.\right.\\
&\qquad\left.\left.+(d-1)(n_{\text{elem}}+p+(2^{\ell}-1)(p-1))^{(d-2)}p\right)^3\right)\\
&+\mathcal{O}\left(\left((n_{\text{elem}}+p+(2^{\ell}-1)(p-1))^{(d-1)}(d+2)\right.\right.\\
&\qquad\left.\left.+(d-1)(n_{\text{elem}}+p+(2^{\ell}-1)(p-1))^{(d-2)}\right)^3\right).
\end{aligned} \tag{6.28}$$

6.4. Problem implementation

We implement the model problem using PetIGA-MF, a multi-field extension of PetIGA that we present in [110]. This framework allows to use a different space for each field of interest. For instance, we can use $\mathbf{H}(div)$ spaces to solve incompressible flows or $\mathbf{H}(curl)$ discretizations to solve electromagnetism problems. PetIGA-MF is based on PETSc [11, 10] and uses the data management libraries to pack the data of the multiple fields in a single object, thus simplifying the discretization construction.

We solve the system that results from the discretization of the problem using the sequential version of the multifrontal solver MUMPS [4, 5] with the automatic choice of partitioning technique, resulting in METIS [84] algorithm for all the cases.

We report the global FLOPs and computational times (in seconds). This allows us to analyze the impact of the local reduction of continuity in the computational cost. For each of the mesh sizes and polynomial degrees that we use to discretize the model problem, we consider a range of cases with a particular number of partition levels that locally reduce the continuity.

All computational tests are solved sequentially on TACC Stampede system. Each node is outfitted with turbo boost 2.7 GHz cores (up to peak 3.5 GHz in turbo mode) and 1TB of memory.

6.5. Numerical results

6.5.1. 2D Example with exact solution

First, we compute the solution of the numerical experiment presented in [29]. It consists of a 2D problem with a manufactured solution allowing to analyze the impact of the continuity reduction on the total error. The problem is solved in a unitary domain $\Omega = (0,1)^2$ with a no-slip condition imposed on its boundary ($\partial\Omega$), and a value of the pressure $p_e = 0$ at the left-bottom corner. The kinematic viscosity is assumed to be one ($\nu = 1$). Two types of flows are studied: a Stokes flow ($\beta_{rec} = 0$) and a Darcy flow ($\beta_{rec} = 1000$). The external volumetric force is equal to

$$\mathbf{f} = \beta_{rec}\bar{\mathbf{u}} - \nabla \cdot (2\nu\nabla^s\bar{\mathbf{u}}) + \nabla\bar{\mathbf{p}}$$

with

$$\bar{\mathbf{u}} = \begin{bmatrix} 2e^x\left(-1+x\right)^2 x^2\left(y^2-y\right)\left(-1+2y\right) \\ e^x\left(-1+x\right)x\left(-2+x\left(3+x\right)\right)\left(-1+y\right)^2 y^2 \end{bmatrix},$$

and

$$\begin{aligned}\bar{p} = -424 + 156e + (y^2-y)(-456 + e^x(456 + x^2(228 - 5(y^2-y)) \\ + 2x(-228 + (y^2-y)) + 2x^3(-36 + (y^2-y)) + x^4(12 + (y^2-y)))).\end{aligned}$$

The domain discretization is defined by the tensor product of unmapped NURBS functions since the domain is a unitary square. We use structured meshes with the same number of elements in each dimension (n_{elem}) and sizes of 128^2, 256^2, 512^2 and 1024^2 elements. We consider polynomial degrees p in a range from 2 to 4. We analyze all rIGA cases starting with the one that involves no continuity reduction at any partition level (that corresponds to traditional IGA), until reaching the case with continuity reduction along all inter-element boundaries.

Table 6.3 shows the L^2 norm of the approximation error of the velocity for the Stokes problem ($\beta_{rec} = 0$) discretized with IGA and the optimal (in computational savings) case of rIGA. The discretization error of the computed solution first improves as we enrich the solution spaces, by enlarging the mesh sizes, increasing the polynomial order and/or reducing continuity. However, due to several factors, such as the rounding errors in the numerical results of the multiple operations performed by the LU factorization, the conditioning error of the system and the machine precision, the minimum error we can achieve for a computed solution is limited. The limit in error is well observed for cases with large mesh sizes and high polynomial degrees.

The upper bound on the approximation error is $\mathcal{O}(\kappa\,\epsilon)$, where κ is a coefficient proportional to the condition number of the system matrix, and ϵ is the machine precision, that is, $\mathcal{O}$(1e-16) for double precision arithmetic. The condition number of systems matrices in discretizations based on B-splines grows with respect to the mesh size, the polynomial order, and the continuity degree [64]. Thus, the approximation error increases, as we observe in the numerical results (Table 6.3).

To eliminate the conditioning errors from the analysis, we employ the iterative refinement method [121, 94]. This approach allows to reach an upper bound on the approximation error

Polynomial degree	Method	Mesh size (N_{elem})			
		128^2	256^2	512^2	1024^2
2	IGA	8.66e-09	1.09e-09	1.36e-10	1.79e-11
	rIGA	8.66e-09	1.09e-09	1.36e-10	1.73e-11
3	IGA	4.02e-11	2.52e-12	7.79e-13	7.27e-13
	rIGA	3.82e-11	2.39e-12	1.48e-12	1.92e-12
4	IGA	1.37e-13	3.98e-14	3.53e-13	7.43e-13
	rIGA	2.83e-13	1.61e-13	1.64e-12	4.46e-12

Table 6.3.: L^2 norm of the approximation error of the velocity for the 2D Stokes problem ($\beta_{rec} = 0$).

of $\mathcal{O}(\epsilon)$, assuming no rounding errors in the direct solver. Table 6.4 shows the L^2 norm of the approximation error of the velocity for the Stokes problem ($\beta_{rec} = 0$) discretized with IGA and the optimal case of rIGA.

Polynomial degree	Method	Mesh size (N_{elem})			
		128^2	256^2	512^2	1024^2
2	IGA	8.66e-09	1.09e-09	1.36e-10	1.70e-11
	rIGA	8.66e-09	1.09e-09	1.36e-10	1.70e-11
3	IGA	4.02e-11	2.52e-12	1.58e-13	1.45e-14
	rIGA	3.83e-11	2.46e-12	1.54e-13	1.36e-14
4	IGA	1.36e-13	4.76e-15	5.65e-15	2.27e-14
	rIGA	1.33e-13	4.28e-15	5.86e-16	6.40e-15

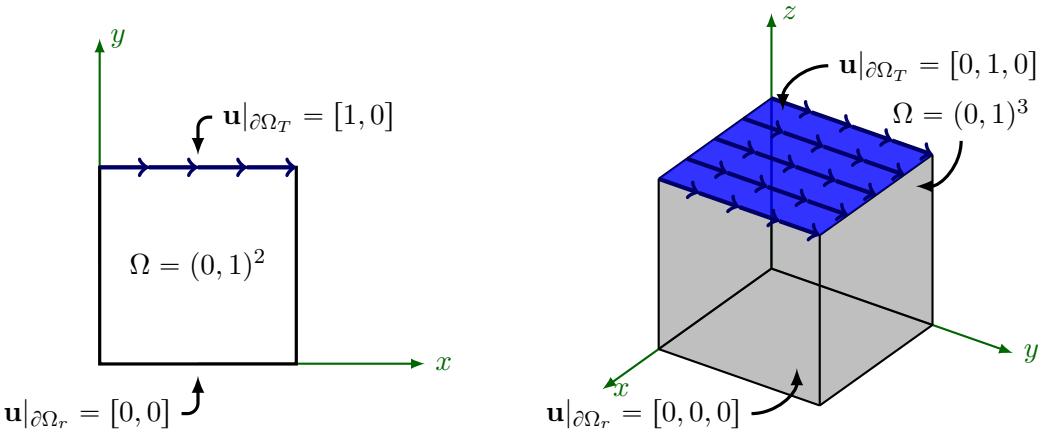
Table 6.4.: L^2 norm of the approximation error of the velocity for the 2D Stokes problem ($\beta_{rec} = 0$). This error is obtained with an iterative refinement method that minimizes the impact of round-off effects.

The numerical results show that rIGA involves an improvement in the discretization error. For Darcy flow ($\beta = 1000$), we obtain a similar error for the velocity than for the Stokes flow ($\beta_{rec} = 0$). Assuming that the system matrix on both flows types involves a comparable condition number, then, the upper bound on the error of the computed solution are close to each other. Additionally, Darcy flow shows the same discretization error than for the Stokes flow, when we use the iterative refinement method (Table 6.4).

6.5.2. Lid-driven cavity problem (Stokes flow)

We solve a lid-driven cavity problem over a unitary domain, assuming a Stokes flow ($\beta_{rec} = 0$). We impose a tangential velocity at the top boundary of the domain ($\partial\Omega_T$) equal to one, and in

the remaining boundaries ($\partial\Omega_r$) we consider a no-slip condition. Moreover, no external forces are considered ($\mathbf{f} = \mathbf{0}$). Figure 6.4 illustrates the structure of the lid-driven cavity test problem.



(a) Domain $\Omega = (0, 1)^2$

(b) Domain $\Omega = (0, 1)^3$.

Figure 6.4.: Lid-driven cavity fluid flow test problem.

The pressure and the stress fields, in this problem, involve corner singularities. Due to this, the exact solution of the velocity lies in a Sobolev space $\mathcal{W}^{1,p}(\Omega)$, being $1 < p \leqslant 2$, instead of lying in $\mathbf{H}(div; \Omega)$ or $\mathbf{H}^1(\Omega)$. In [58], Evans et al. showed that the IGA discretization presented in subsection 6.2 approximates well the smooth portions of the flow and properly resolve the flow close to the corner singularities considering that the vorticity close to those corners slowly converges to the highly accurate pseudospectral results presented in [28].

In addition to this classical Stokes benchmark problem, we consider a 2D case in which the boundary condition at the top boundary of the domain consists of a continuous function

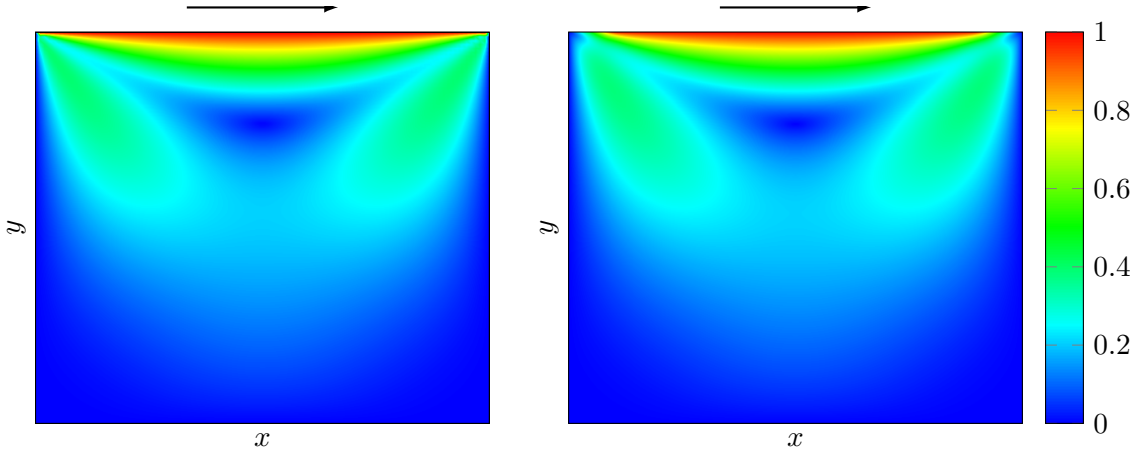
$$\mathbf{u}|_{\partial\Omega_T} = \begin{cases} [10x, 0], & \forall\, x < 0.1 \\ [1, 0]. & \forall\, 0.1 \leqslant x \leqslant 0.9 \\ [10(1 - x), 0]. & \forall\, 0.9 \leqslant x \leqslant 1 \end{cases} \tag{6.29}$$

In this case, the problem does not experience corner singularities, and the exact solution of the fluid flow problem is in the proper spaces.

These model problems are discretizedd by the tensor-product of unmapped NURBS functions resulting in meshes with the same number of elements per side (n_{elem}). We employ mesh sizes of 512^2 and 1024^2 elements in 2D, while in 3D, we use mesh sizes of 16^3 and 32^3 elements. Moreover, we consider polynomial degrees p from 2 to 5. These polynomial orders are kept constant for each problem. To analyze the impact of continuity reduction on the computational time, we consider all range of cases with a particular number of partition levels that locally reduce continuity.

Figure 6.5 illustrates the magnitude of the velocity for the Stokes problem solved using a mesh size of 1024^2 elements, and polynomial degree $p = 4$ in 2D. In Figure 6.6, we plot the horizontal

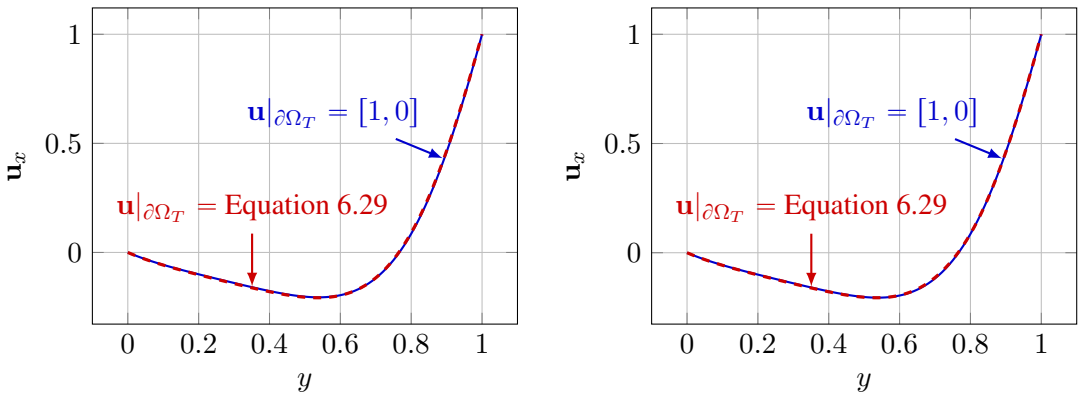
velocity along the vertical centerline when using the top boundary condition $\mathbf{u}|_{\partial\Omega_T} = [1, 0]$ and the one defined with Equation 6.29, and Figure 6.7 compares the horizontal velocity along the vertical centerline for IGA and the optimal case of rIGA in both 2D and 3D with $\mathbf{u}|_{\partial\Omega_T} = [1, 0]$ (that corresponds to the classical Stokes benchmark problem). The smooth portion of the flow inside the domain is accurately approximated, and the continuity reduction shows no notorious impact on the solution, as expected.



(a) Case with the top boundary condition of $\mathbf{u}|_{\partial\Omega_T} = [1, 0]$.

(b) Case with the top boundary condition defined by Equation 6.29.

Figure 6.5.: Magnitude of the velocity ($\mathbf{u}$) of the 2D Stokes problem. Problem solution approximated over a mesh size of 1024^2 elements and a polynomial degree $p = 4$.

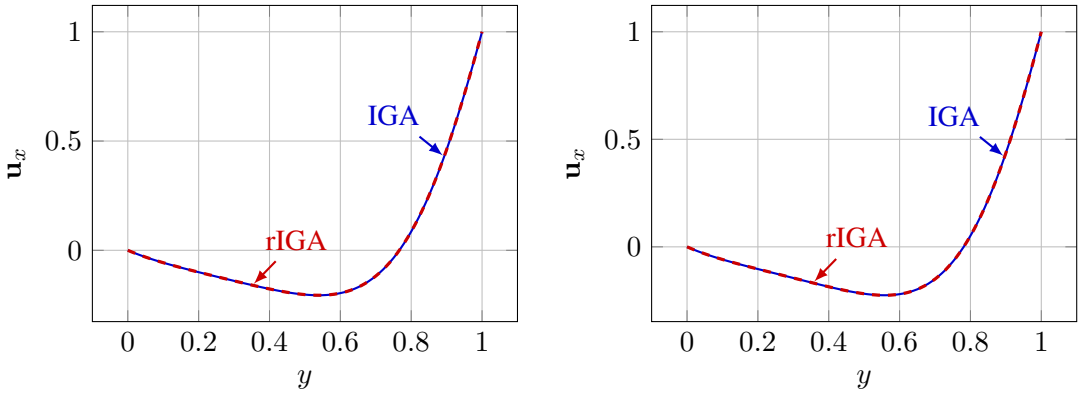


(a) IGA

(b) rIGA

Figure 6.6.: Horizontal velocity $\mathbf{u}_x$ along the vertical centerline. Comparison between using a top boundary condition of $\mathbf{u}|_{\partial\Omega_T} = [1, 0]$ and $\mathbf{u}|_{\partial\Omega_T}$ defined by Equation 6.29.

Figures 6.8 and 6.9 display the number of FLOPs required to solve the Stokes problem in 2D and 3D, respectively. The number of FLOPs is plotted with respect to the macro-elements size

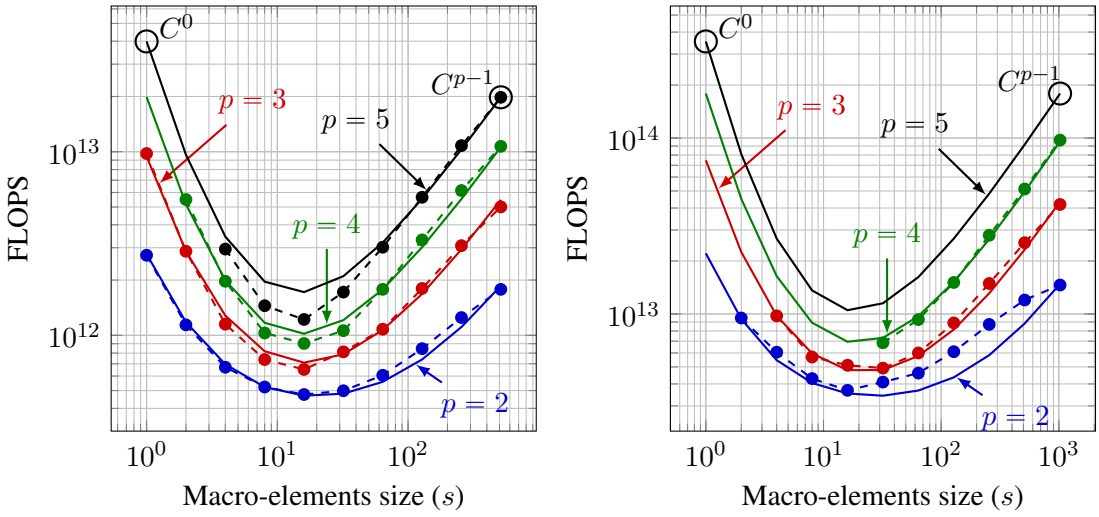


(a) 2D: Mesh size of 1024^2 elements and $p = 4$

(b) 3D: Mesh size of 32^2 elements and $p = 4$

Figure 6.7.: Comparison of the horizontal velocity $\mathbf{u}_x$ along the vertical centerline for IGA and the optimal case of rIGA.

$s = n_{\text{elem}}/2^{\ell}$, being ℓ the number of partition levels that perform continuity reduction.

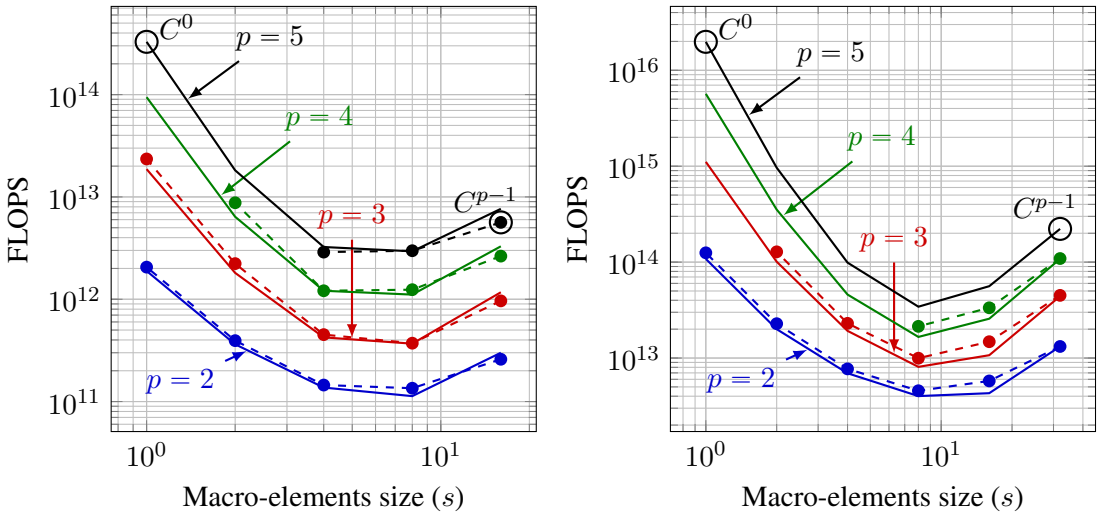


(a) Mesh size of 512^2 elements

(b) Mesh size of 1024^2 elements

Figure 6.8.: Number of FLOPs required to solve the 2D Stokes problem with the multifrontal direct solver. The dashed lines with rounded markers (- ●-) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

For 2D, we use a constant factor close to 16 to fit the theoretical estimates provided in Equation 6.15 and 6.28 with the computed number of FLOPs. For 3D, the constant factor used to fit the theoretical estimates with the numerical results is close to 7. These constants include the contribution of forming the Schur complements and the total number of FLOPs that LAPACK



(a) Mesh size of 16^3 elements

(b) Mesh size of 32^3 elements

Figure 6.9.: Number of FLOPs required to solve the 3D Stokes problem with the multifrontal direct solver. The dashed lines with rounded markers (- ● -) correspond to the numerical results and the solid lines (——) represent the theoretical estimates.

performs to factorize the system, and are slightly dependent on the polynomial degree.

Tables 6.5 and 6.6 provide the number of FLOPs and computational times (in seconds) for the corresponding IGA and optimal rIGA discretizations in 2D and 3D, respectively.

Polynomial degree	Method	Mesh size (N_{elem})			
		512^2		1024^2	
		FLOPs	Time $[s]$	FLOPs	Time $[s]$
2	IGA	1.78e+12	106.27	1.45e+13	794.28
	rIGA	4.76e+11	34.51	3.68e+12	226.29
3	IGA	5.01e+12	277.83	4.19e+13	2191.69
	rIGA	6.52e+11	46.24	4.92e+12	309.11
4	IGA	1.07e+13	583.97	9.73e+13	4975.04
	rIGA	9.03e+11	62.67	6.85e+12	429.88
5	IGA	1.98e+13	1038.72	1.93e+14	12996.36
	rIGA	1.24e+12	83.32	8.04e+12	645.24

Note: Results in blue were computed with Pardiso and PETSc using 64bit indices.

Table 6.5.: Number of FLOPs and computational times (in seconds) required to solve the 2D Stokes problem with the multifrontal direct solver using two mesh sizes and four polynomial degrees (p ranging from 2 to 5).

Polynomial degree	Method	Mesh size (N_{elem})			
		16^3		32^3	
		FLOPs	Time $[s]$	FLOPs	Time $[s]$
2	IGA	2.59e+11	15.36	1.32e+13	667.42
	rIGA	1.34e+11	8.79	4.56e+12	242.99
3	IGA	9.63e+11	53.42	4.49e+13	2222.29
	rIGA	3.72e+11	22.78	9.96e+12	530.63
4	IGA	2.63e+12	138.27	1.09e+14	5394.60
	rIGA	1.24e+12	69.58	2.14e+13	1179.58
5	IGA	5.64e+12	285.07	2.26e+14	14674.84
	rIGA	2.89e+12	165.49	4.50e+13	3019.07

Note: Results in blue were computed with Pardiso and PETSc using 64bit indices.

Table 6.6.: Number of FLOPs and computational times (in seconds) required to solve the 3D Stokes problem with the multifrontal direct solver using two mesh sizes and four polynomial degrees (p ranging from 2 to 5).

The theoretical estimates approximate well the numerical results. In particular, these estimates predict the macro-element size that delivers the maximum cost reduction. The numerical results show that rIGA decreases the number of FLOPs required for solving the Stokes problem by a factor up to $\mathcal{O}(p^2)$ in 2D. For instance, in the case that we solve the model problem using a mesh size of 1024^2 elements and polynomial degree $p = 4$, the optimal rIGA case reports a reduction with respect to C^{p-1} IGA of approximately 14 times in terms of the number of FLOPs. In 3D, the reduction factor of the number of FLOPs is approximately p. In 3D, the examples we can solve are in the pre-asymptotic regime, and due to that, the maximum reduction factor of the number of FLOPs we can observe is $\mathcal{O}(p)$. As we reach the asymptotic regime (for a sufficiently large grid), the reduction factor of the number of FLOPs becomes $\mathcal{O}(p^2)$. For example, when using the FLOPs estimates of Equation 6.28 with $p = 5$ and $N_{\text{elem}} = 256^3$ we obtain a saving factor of 35.

In terms of computational times, rIGA reports a maximum reduction factor with respect to C^{p-1} IGA of almost 12 times when solving the Stokes problem in 2D, with a mesh size of 1024^2 elements and polynomial degree $p = 4$. The problem discretized with the optimal rIGA case requires approximately seven minutes to be solved, while in case of IGA, the model problem requires 83 minutes (almost one and a half hours) to be solved. In 3D, the case solved with a mesh size of 32^3 elements and polynomial degree $p = 4$ reports the largest solvable problem using sequential MUMPS. In this case, the factor is equal to 4.57: from 90 (with IGA) minutes to 20 (with rIGA).

7. Conclusions and Future work

7.1. Conclusions

This dissertation proposes a refined Isogeometric Analysis (rIGA) method to solve problems governed by Partial Differential Equations (PDEs). Starting from a highly continuous Isogeometric Analysis (IGA) discretization, our strategy reduces the continuity over certain hyperplanes that split the mesh into subdomains (or macro-elements). Considering that the hyperplanes act as separators during the elimination of Degrees of Freedom (DoF) on direct solvers, we developed a method that lowers the continuity until reaching a zero degree (C^0) over selected hyperplanes (rIGA), and an Optimally refined Isogeometric Analysis (OrIGA) that lowers the continuity until arbitrary degrees (C^k).

Additionally, we also constructed a hybrid solver strategy that applies rIGA to iterative solvers. This approach reduces the continuity until degree zero (C^0) over the inter-subdomains boundaries, and then it performs a static condensation at the macro-elements level. The skeleton system that results from assembling the Schur complement of all the macro-elements is solved using an iterative strategy.

To illustrate the impact of the continuity reduction on the computational cost, we report the FLoating Point Operations (FLOPs) and computational times to solve linear systems resulting from the rIGA discretization with structured meshes and uniform polynomial orders. In particular, we analyze the computational cost for the cases of: (a) a Laplace problem in 2D and 3D solved with the original rIGA,(see Section 5.1), (b) a 2D Laplace problem solved using OrIGA,(see Section 5.2), (c) a 2D Poisson problem with a forcing based on sines that is solved with the hybrid solver strategy (see Section 5.3), and (d) a fluid mechanics application of incompressible fluid flow in a bounded domain (see Section 6.5). Neither traditional Finite Element Analysis (FEA) nor IGA provides the optimal number of FLOPs for solving those problems with a fixed mesh size. The optimal is achieved by the rIGA methods we introduce herein.

7.1.1. rIGA

The original version of rIGA for direct solvers computes the solution of problems (with large mesh sizes) approximately p^2 times faster than C^{p-1} IGA, and the gains with respect to FEA are even larger, especially in 3D. Moreover, the memory requirements also decrease significantly, as observed in section 5.1.4.3.

In 2D, the cases solved using macro-elements (subdomains) composed of 16^2 elements report the lowest computational times in most of the tested configurations, while in 3D, a wider range of macro-element size options is observed. For small mesh sizes ($N_{\text{elem}} = 32^3$), optimal macro-elements consist of 8^3 elements, while as the mesh size increases to $N_{\text{elem}} = 128^3$, the optimal macro-element size tends to 16^3 elements.

The maximum polynomial order that we use to solve the model problem in 2D is $p = 9$. For this polynomial degree, we were not able to compute the optimal rIGA case that consists of a discretization with macro-elements of size $s = 16^2$ due to the limit on the number of NonZero (NZ) entries that PETSc configured with MUMPS solver has. The best results we obtain correspond to a discretization with macro-elements of size $s = 32^2$. This case reports a gain factor with respect to IGA (C^{p-1}) of almost 70 times in terms of FLOPs and more than 35 times in terms of time. For instance, in 2D, the problem that we solve in 2 hours with IGA, requires approximately 3 minutes to be solved with rIGA. Theoretical estimates predict that these savings will further increase as we consider larger problems with higher p.

In 3D, the maximum reproducible gain with respect to C^{p-1} is 13.69, which corresponds to the case with mesh size $N_{\text{elem}} = 128^3$ and polynomial order $p = 3$. In this case, the problem that we solve in one hour with rIGA requires approximately 15 hours to be solved with IGA, and more than 100 hours if one employs FEA.

7.1.2. OrIGA

The extended version of rIGA, called OrIGA, leads to systems of linear equations that can be more efficiently solved with direct solvers than those obtained with the original version of rIGA. In 2D, OrIGA reduces the total computational cost needed by the direct solver to compute the solution of the model problem by up to 25% (for large mesh sizes and polynomial degrees) when compared to the original version of rIGA. When we compare to C^{p-1} IGA and C^0 FEA, OrIGA reports a reduction in the number of FLOPs of a factor up to 55.

7.1.3. Hybrid solver strategy with rIGA

The hybrid solver strategy delivers moderate savings in terms of computational time when solving the 2D Poisson model problem. These savings are smaller than those obtained using rIGA with direct solvers. For instance, in a mesh with 2048^2 elements and polynomial degree $p = 5$, the hybrid solver strategy solves the model problem approximately 2.37 times faster than with C^{p-1} IGA, while with direct solvers, the original version of rIGA reduces the computational time by a factor of approximately 22 with respect to C^{p-1} IGA [67] (see Section 5.1).

The cost of all operations in the hybrid solver strategy is linear with respect to the number of elements, except for the case of the number of iterations, which grows as $n_{\text{elem}}^{\alpha d}$, with $\alpha > 1$. For the 2D case, the computational savings of rIGA increase as compared to IGA up to an asymptotic limit. For instance, the model problem discretized with a mesh of size of $N_{\text{elem}} = 4096^2$ and polynomial degree $p = 3$ delivers a reduction factor of approximately 3. This is larger than when using a mesh size of $N_{\text{elem}} = 2048^2$ elements with the same polynomial degree which involves a reduction factor of approximately 2.

Comparing the performance of the iterative solver when applied to rIGA discretizations vs when applied to C^{p-1} IGA, we observe a larger reduction in terms of time than in terms of the number of FLOPs. This may be a result of the domain partitioning, which simplifies the data transfers with respect to the bandwidth limitations of the machine. Large sparse systems (as in C^{p-1} IGA) degrade the performance of the memory access and the data transmission. In those cases, the machine often fails to optimally fill the cache memory resulting in a delay of some

operations until the required data is transferred. The division of the problem in subproblems (macro-elements) improves the performance of the cache-based machines. The solution of the subproblems delivers a smaller global system that not only better exploits the data locality but also benefits in the filling of the cache [51].

However, when the macro-elements are small ($s \sim p$), the cost to access the memory becomes significant with respect to the cost to operate on the data (eliminate DoF) [51]. This memory access impacts the scaling of the computational time of the static condensation (Figure 5.14). Once the size of the macro-elements is large enough ($s \gg p$), data operations dominate the timings. At this point, the reduction factor of computational time corresponds to that of the number of FLOPs.

In 3D, the number of NZ entries of the rIGA skeleton matrix is larger than that of the C^{p-1} IGA matrix. This results in a significant increment in the relative cost of matrix-vector multiplication. Thus, rIGA combined with the hybrid solver strategy delivers no savings in the computational cost when solving 3D problems, and its use is not advised in this case.

7.1.4. Applications

The implementation of rIGA to solve incompressible fluid flow problems delivers a reduction in the computational cost, and it provides better accuracy than C^{p-1} IGA. In 2D, the numerical results show a reduction factor in the computational cost up to p^2. In 3D, the maximum reproducible problems with sequential MUMPS are in the pre-asymptotic regime, and the maximum gain factors are of $\mathcal{O}(p)$. In multi-field problems, we require more time (larger grids than in scalar problems) to arrive at the asymptotic limit and reach the full savings because the system is bigger (more equations). For sufficiently large grids (asymptotic regime), the theoretical estimates show that the gain factor becomes $\mathcal{O}(p^2)$.

The optimal discretizations obtained with rIGA consists of enriched/nested spaces with respect to C^{p-1} IGA. Therefore, the best approximation error is improved by definition. Similarly, the best approximation error of the corresponding C^0 FEA discretization is smaller than that of rIGA. The total numerical error for stable elliptic problems improves when going from IGA to rIGA discretizations. However, for the case of hyperbolic and parabolic problems, stability may play a crucial role on the total approximation error. Therefore, using rIGA not necessarily implies a direct improvement of the results and it may even lead to a worse result even if rIGA reduces the best approximation error.

7.2. Future work

In the near future, we plan to apply rIGA to solve hyperbolic and parabolic PDEs systems in order to perform deeper analysis on the effect of the continuity reduction on the total numerical error. In particular, we shall evaluate how significant is the impact of the continuity reduction on the problems' stability. As a remark, the first work in this topic has been just published this year. In this work, the authors study the spectral approximation properties of rIGA and that show how the local reduction of continuity impact on the error in the eigenvalues and eigenfunctions [104].

As another line of future research, we plan to implement non-tensor product rIGA discretizations by using T-splines. The savings for this strategy may vary depending upon the particular discretization. The main idea is to employ C^0 T-splines on the top level of a given macro-element in order to *separate* the interior of the possibly locally-refined macro-element from the rest of the computational domain.

We are also planning to work on a parallel implementation of the rIGA. The use of rIGA is expected to be beneficial in terms of distributed memory parallel computations, since separators diminish the amount of information that needs to be shared among neighboring processors, thus, minimizing the communication cost and increasing its parallel scalability with respect to IGA.

Furthermore, we plan on using the hybrid solver method with the Conjugate Gradients (CG) iterative solver preconditioned with the Balancing Domain Decomposition by Constraints (BDDC) strategy [89, 90, 9]. By using this preconditioner technique, it is possible to represent the skeleton system in a reduced form that has fewer NZ entries. This may result in larger computational savings for 2D, and it will permit to use hybrid solver strategy in 3D, considering that the main limitation in 3D is that the skeleton system involves a larger number of NZ entries than the original IGA system.

8. Main achievements

8.1. Peer reviewed publications

2018 D. Garcia, D. Pardo, L. Dalcin, and V.M. Calo. ***Refined Isogeometric Analysis for a Preconditioned Conjugate Gradient Solver.*** Computer Methods in Applied Mechanics and Engineering (in press), 2018.
https://doi.org/10.1016/j.cma.2018.02.006.

2017 D. Garcia, M. Bartoň, and D. Pardo. ***Optimally refined isogeometric analysis.*** Procedia Computer Science, 108:808 – 817, 2017.
https://doi.org/10.1016/j.procs.2017.05.283.

2017 D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V.M. Calo. ***The value of continuity: Refined isogeometric analysis and fast direct solvers.*** Computer Methods in Applied Mechanics and Engineering, 316: 586 – 605, 2017.
https://doi.org/10.1016/j.cma.2016.08.017.

8.2. Conferences talks

2018 D. Garcia, D. Pardo, V.M. Calo and J. Muñoz-Matute. ***refined Isogeometric Analysis (rIGA): A multi-field application on a fluid flow scenario.*** ICCS, Wuxi, China.

2017 D. Garcia, M. Bartoň, and D. Pardo. ***Optimally refined isogeometric analysis.*** ICCS, Zurich, Switzerland.

2016 D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V.M. Calo. ***Refined Isogeometric Analysis: Improved Performance of Direct Solvers by Controlling Continuity.*** WCCM XII & APCOM VI, Seoul, South Korea.

2016 D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V.M. Calo. ***Refined Isogeometric Analysis (rIGA)*** HOFEIM 2016, Jerusalem, Israel.

2016 M. Paszyński, G. Gurgul, D. Garcia, and D. Pardo. ***Efficient parallelization of direct solvers for isogeometric analysis*** PMAA 2016, Bordeaux, France.

8.3. Seminars & Workshops

2018 D. Garcia, D. Pardo, and V.M. Calo. ***Refined Isogeometric Analysis (rIGA) with multi-physics applications.*** Workshop: "Fifth International Congress On Multiphysics, Multi-scale, and Optimization Problems", BCAM, Bilbao, Spain.

2016 D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, and V.M. Calo. ***Refined Isogeometric Analysis: Improved Performance of Direct Solvers by Controlling Continuity.*** Workshop: "Fourth International Congress On Multiphysics, Multiscale, and Optimization Problems", BCAM, Bilbao, Spain.

2016 D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V.M. Calo. ***Refined Isogeometric Analysis: Complexity Reduction of Direct Solvers by Controlled Continuity.*** Seminar at Institute for computational Engineering and Sciences (ICES), The University of Texas at Austin, USA.

2016 D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V.M. Calo. ***Refined Isogeometric Analysis.*** Workshop: "The Sixth Valparaíso's Mathematics and Applications Days" (V-MAD 6), Valparaiso, Chile.

2014 D. Garcia, V.M. Calo, and D. Pardo. ***k-Refinement on 1D Problems.*** Workshop: "Third International Workshop On Multiphysics, Multiscale, and Optimization Problems", BCAM, Bilbao, Spain.

A. Memory estimates for refined Isogeometric Analysis (rIGA) with direct solvers

In order to estimate the memory requirements to solve a numerical problem with rIGA, we compute the number of NonZero (NZ) entries in factors L and U based on the matrix decomposition procedure performed by the multifrontal direct solver. The matrix decomposition eliminates the Degrees of Freedom (DoF) in sets (either subsystems or separators). Each set consists of q fully assembled DoF and its elimination results in $\mathcal{O}\left(q^2\right)$ NZ entries. Therefore, the total number of NZ entries on the factors L and U is given by

$$\mathcal{O}\left(\Psi|_{C^0}\left(q_{\text{sep}}|_{C^0}\right)^2\right) + \eta_{\text{m-e}}\left(\mathcal{O}\left(\Psi|_{C^{p-1}}\left(q_{\text{m-e}}\right)^2\right)\right),$$

$$\mathcal{O}\left(\Psi|_{C^0}\left(N^{(d-1)/d}\right)^2\right) + 2^{id}\,\mathcal{O}\left(\Psi|_{C^{p-1}}\left(\left(\frac{N}{2^{id}}\right)^{(d-1)/d} p\right)^2\right),$$

$$\mathcal{O}\left(\Psi|_{C^0}\ n^{2(d-1)}\right) + \frac{2^{id}}{2^{2i(d-1)}}\,\mathcal{O}\left(\Psi|_{C^{p-1}}\ n^{2(d-1)}p^2\right) + L.O.T.,$$

where 2^{id} and $N/2^{id}$ are the number and size of the macro-elements after i partition levels. $\Psi|_{C^0}$ and $\Psi|_{C^{p-1}}$ variables include the contribution of all the separators to the total number of NZ entries. We compute $\Psi|_{C^0}$ as

$$\Psi|_{C^0} = \sum_{j=0}^{i-1}\sum_{m=0}^{d-1} 2^{dj+m}\left(2^{-(j(d-1)+m)}\right)^2 = \begin{cases} \frac{3}{2}\left(\log_2\left(2^i\right)\right), & 2D \\ \frac{7}{2}\left(1-2^{-i}\right), & 3D \end{cases}$$

where j refers to the j-th level of partition, and $j+m$ is the number of partitions per level. In 2D, we perform the same number of partitions along the horizontal and vertical dimensions on the mesh. We now assume that $\Psi|_{C^{p-1}}$ behaves as

$$\Psi|_{C^{p-1}} = \begin{cases} \frac{3}{2}\left(\log_2\left(2^\zeta\right)\right)^2, & 2D \\ \frac{7}{2}\left(1-2^{-\zeta}\right) \approx \frac{7}{2}, & 3D \end{cases}$$

where ζ is the number of levels of the macro-elements partitioning. These levels correspond to the ones involving C^{p-1} separators. Finally, the total number of NZ entries for 2D and 3D are

$$2D: \quad \chi = \mathcal{O}\left(\left(\underbrace{\frac{3}{2}\log_2\left(2^{-i}\right)}_{\substack{C^0\text{-separators}\\ \text{contribution}}} + \underbrace{\frac{3}{2}\log_2\left(2^{\zeta}\right)^2 p^2}_{\substack{C^{p-1}\text{ macro-elements}\\ \text{contribution}}}\right) n^2\right) + L.O.T.\,,$$

$$3D: \quad \chi = \mathcal{O}\left(\left(\underbrace{\frac{7}{2}\left(1-2^{-i}\right)}_{\substack{C^0\text{-separators}\\ \text{contribution}}} + \underbrace{\frac{7}{2}2^{-i}\,p^2}_{\substack{C^{p-1}\text{ macro-elements}\\ \text{contribution}}}\right) n^4\right) + L.O.T.$$

These equations are multiplied by $8 \cdot 10^{-6}$ to express the memory estimation in $Mbytes$. Factor 8 accounts for computations performed in double precision.

Bibliography

[1] S. Adams and B. Cockburn. A Mixed Finite Element Method for Elasticity in Three Dimensions. Journal of Scientific Computing, 25(3):515–521, 2005. (cited in page(s) v, 1)

[2] M. Ainsworth, P. Davies, D. Duncan, P. Martin, and B. Rynne. Topics in Computational Wave Propagation: Direct and Inverse Problems. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2012. (cited in page(s) v, 1)

[3] I. Akkerman, Y. Bazilevs, V.M. Calo, TJR. Hughes, and S. Hulshoff. The role of continuity in residual-based variational multiscale modeling of turbulence. Computational Mechanics, 41(3):371–378, 2008. (cited in page(s) vi, 2)

[4] P. R. Amestoy, I. S. Duff, J.-Y. LÉxcellent, and J. Koster. A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. SIAM Journal on Matrix Analysis and Applications, 23(1):15–41, 2001. (cited in page(s) 46, 49, 86)

[5] P. R. Amestoy, A. Guermouche, J.-Y. L'Excellent, and S. Pralet. Hybrid scheduling for the parallel solution of linear systems. Parallel Computing, 32(2):136 – 156, 2006. (cited in page(s) 46, 49, 86)

[6] H. R. Atri and S. Shojaee. Free Vibration Analysis of Thin-Shell Structures using Finite Element based on Isogeometric Approach. Iranian Journal of Science and Technology, Transactions of Civil Engineering, 40(2):85–96, 2016. (cited in page(s) vi, 2)

[7] F. Auricchio, L. B. da Veiga, A. Buffa, C. Lovadina, A. Reali, and G. Sangalli. A fully "locking-free" isogeometric approach for plane linear elasticity problems: A stream function formulation. Computer Methods in Applied Mechanics and Engineering, 197(1-4):160 – 172, 2007. (cited in page(s) vi, vii, 3)

[8] O. Axclsson and P. S. Vassilevski. Algebraic multilevel preconditioning methods. i. Numerische Mathematik, 56(2):157–177, 1989. (cited in page(s) 19)

[9] S. Badia, A. F. Martín, and J. Principe. Implementation and Scalability Analysis of Balancing Domain Decomposition Methods. Archives of Computational Methods in Engineering, 20(3):239–262, 2013. (cited in page(s) 97)

[10] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016. (cited in page(s) 46, 86)

[11] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Web page, 2016. (cited in page(s) 46, 86)

[12] Y. Bazilevs and I. Akkerman. Large eddy simulation of turbulent Taylor-Couette flow using isogeometric analysis and the residual-based variational multiscale method. Journal of Computational Physics, 229(9):3402 – 3414, 2010. (cited in page(s) vi, 2)

[13] Y. Bazilevs, L. Beirão da Veiga, J. A. Cottrell, TJR. Hughes, and G. Sangalli. Isogeometric Analysis: Approximation, Stability and Error estimates for h-refined meshes. Mathematical Models and Methods in Applied Sciences, 16(07):1031–1090, 2006. (cited in page(s) vii, 3)

[14] Y. Bazilevs, V.M. Calo, J. A. Cottrell, TJR. Hughes, A. Reali, and G. Scovazzi. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Computer Methods in Applied Mechanics and Engineering, 197(1-4):173 – 201, 2007. (cited in page(s) vi, 2)

[15] Y. Bazilevs, V.M. Calo, TJR. Hughes, and Y. Zhang. Isogeometric fluid-structure interaction: theory, algorithms, and computations. Computational Mechanics, 43(1):3–37, 2008. (cited in page(s) vi, 2)

[16] Y. Bazilevs, V.M. Calo, Y. Zhang, and TJR. Hughes. Isogeometric Fluid-structure Interaction Analysis with Applications to Arterial Blood Flow. Computational Mechanics, 38(4-5):310–322, 2006. (cited in page(s) vi, 2)

[17] Y. Bazilevs, J. Gohean, TJR. Hughes, R. Moser, and Y. Zhang. Patient-specific isogeometric fluid–structure interaction analysis of thoracic aortic blood flow due to implantation of the Jarvik 2000 left ventricular assist device. Computer Methods in Applied Mechanics and Engineering, 198(45-46):3534 – 3550, 2009. (cited in page(s) vii, 3)

[18] Y. Bazilevs, C. Michler, V.M. Calo, and TJR. Hughes. Weak Dirichlet boundary conditions for wall-bounded turbulent flows. Computer Methods in Applied Mechanics and Engineering, 196(49-52):4853–4862, 2007. (cited in page(s) vi, 2, 80)

[19] Y. Bazilevs, C. Michler, V.M. Calo, and TJR. Hughes. Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes. Computer Methods in Applied Mechanics and Engineering, 199(13-16):780 – 790, 2010. (cited in page(s) vi, 2)

[20] D. Benson, Y. Bazilevs, M.-C. Hsu, and TJR. Hughes. Isogeometric shell analysis: the Reissner-Mindlin shell. Computer Methods in Applied Mechanics and Engineering, 199(5):276–289, 2010. (cited in page(s) vi, 2)

[21] D. Benson, Y. Bazilevs, M.-C. Hsu, and TJR. Hughes. A large deformation, rotation-free, isogeometric shell. Computer Methods in Applied Mechanics and Engineering, 200(13):1367–1378, 2011. (cited in page(s) vi, 2)

[22] D. Benson, S. Hartmann, Y. Bazilevs, M.-C. Hsu, and TJR. Hughes. Blended isogeometric shells. Computer Methods in Applied Mechanics and Engineering, 255:133–146, 2013. (cited in page(s) vi, 2)

[23] M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. J. Comput. Phys., 182(2):418–477, 2002. (cited in page(s) 19)

[24] M. Benzi, C. D. Meyer, and M. Tůma. A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method. SIAM Journal on Scientific Computing, 17(5):1135–1149, 1996. (cited in page(s) 19)

[25] L. Bernal, V.M. Calo, N. Collier, G. Espinosa, F. Fuentes, and J. Mahecha. Isogeometric Analysis of Hyperelastic Materials Using PetIGA. Procedia Computer Science, 18:1604 – 1613, 2013. (cited in page(s) 46)

[26] L. M. Bernal, V.M. Calo, N. Collier, G. A. Espinosa, F. Fuentes, and J. C. Mahecha. Isogeometric Analysis of Hyperelastic Materials Using PetIGA. Procedia Computer Science, 18:1604 – 1613, 2013. (cited in page(s) vii, 3, 46)

[27] S. Blackford and J. Dongarra. LAPACK Working Note 41 "Installation Guide for LAPACK". Department of Computer Science, University of Tennessee, June 1999. (cited in page(s) 48, 64)

[28] O. Botella and R. Peyret. Benchmark spectral results on the lid-driven cavity flow. Computers & Fluids, 27(4):421–433, 1998. (cited in page(s) 89)

[29] A. Buffa, C. de Falco, and G. Sangalli. Isogeometric Analysis: Stable elements for the 2D Stokes equation. International Journal for Numerical Methods in Fluids, 65(11-12):1407–1422, 2011. (cited in page(s) vi, 2, 78, 87)

[30] A. Buffa, G. Sangalli, and R. Vázquez. Isogeometric analysis in electromagnetics: B-splines approximation. Computer Methods in Applied Mechanics and Engineering, 199(17-20):1143 – 1152, 2010. (cited in page(s) vii, 3, 78)

[31] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young. Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. SIAM Journal on Scientific Computing, 19(1):246–265, 1998. (cited in page(s) 19)

[32] V.M. Calo, N. F. Brasher, Y. Bazilevs, and TJR. Hughes. Multiphysics model for blood flow and drug transport with application to patient-specific coronary artery flow. Computational Mechanics, 43(1):161–177, 2008. (cited in page(s) vii, 3)

[33] V.M. Calo, N. O. Collier, D. Pardo, and M. R. Paszyński. Computational complexity and memory usage for multi-frontal direct solvers used in p finite element analysis. Procedia Computer Science, 4(0):1854 – 1861, 2011. (cited in page(s) vii, 3, 18, 25, 46)

[34] K. Chang, TJR. Hughes, and V.M. Calo. Isogeometric variational multiscale large-eddy simulation of fully-developed turbulent flow over a wavy wall. Computers & Fluids, 68:94–104, 2012. (cited in page(s) vi, 2)

[35] K. Chen. Matrix preconditioning techniques and applications, volume 19. Cambridge University Press, 2005. (cited in page(s) 19)

[36] E. Chow and Y. Saad. Approximate Inverse Preconditioners via Sparse-Sparse Iterations. SIAM Journal on Scientific Computing, 19(3):995–1023, 1998. (cited in page(s) 19)

[37] B. Cockburn and J. Gopalakrishnan. Incompressible Finite Elements via Hybridization. Part I: The Stokes System in Two Space Dimensions. SIAM Journal on Numerical Analysis, 43(4):1627–1650, 2005. (cited in page(s) v, 1)

[38] N. Collier, L. Dalcin, and V.M. Calo. On the computational efficiency of isogeometric methods for smooth elliptic problems using direct solvers. International Journal for Numerical Methods in Engineering, 100(8):620–632, 2014. (cited in page(s) vii, 3, 25, 27, 46, 80)

[39] N. Collier, L. Dalcin, D. Pardo, and V.M. Calo. The cost of continuity: Performance of Iterative Solvers on Isogeometric Finite Elements. SIAM Journal on Scientific Computing, 35(2):A767–A784, 2013. (cited in page(s) viii, 4, 43, 44, 46)

[40] N. Collier, L. Dalcin, D. Pardo, and V.M. Calo. The cost of continuity: Performance of iterative solvers on isogeometric finite elements. SIAM Journal on Scientific Computing, 35(2):A767–A784, 2013. (cited in page(s) 20, 21)

[41] N. Collier, D. Pardo, L. Dalcin, M. Paszyński, and V.M. Calo. The cost of continuity: A study of the performance of isogeometric finite elements using direct solvers. Computer Methods in Applied Mechanics and Engineering, 213-216(0):353 – 361, 2012. (cited in page(s) vii, 3, 19, 25, 29, 32, 46, 80)

[42] A. Côrtes, A. Coutinho, L. Dalcin, and V.M. Calo. Performance evaluation of block-diagonal preconditioners for the divergence-conforming B-spline discretization of the Stokes system. Journal of Computational Science, 11:123 – 136, 2015. (cited in page(s) 46)

[43] J. A. Cottrell, TJR. Hughes, and Y. Bazilevs. Isogeometric Analysis: Toward Integration of CAD and FEA. John Wiley & Sons, Ltd, Singapore, Ma, 2009. (cited in page(s) vi, 2, 7, 9, 13)

[44] J. A. Cottrell, TJR. Hughes, and A. Reali. Studies of refinement and continuity in isogeometric structural analysis. Computer Methods in Applied Mechanics and Engineering, 196(41–44):4160 – 4183, 2007. (cited in page(s) vii, 3, 7, 9)

[45] J. A. Cottrell, A. Reali, Y. Bazilevs, and TJR. Hughes. Isogeometric analysis of structural vibrations. Computer Methods in Applied Mechanics and Engineering, 195(41-43):5257 – 5296, 2006. (cited in page(s) vi, vii, 3)

[46] M. G. Cox. The numerical evaluation of B-splines. IMA Journal of Applied Mathematics, 10(2):134–149, 1972. (cited in page(s) 7)

[47] L. B. da Veiga, A. Buffa, C. Lovadina, M. Martinelli, and G. Sangalli. An isogeometric method for the Reissner-Mindlin plate bending problem. Computer Methods in Applied Mechanics and Engineering, 209:45–53, 2012. (cited in page(s) vi, 2)

[48] L. Dalcin, N. Collier, P. Vignal, A. Côrtes, and V.M. Calo. PetIGA: A framework for high-performance isogeometric analysis. Computer Methods in Applied Mechanics and Engineering, 308:151–181, 2016. (cited in page(s) 46)

[49] C. de Boor. On calculating with B-splines. Journal of Approximation Theory, 6(1):50 – 62, 1972. (cited in page(s) 7)

[50] L. Demkowicz. Finite Element Methods for Maxwell Equations. John Wiley & Sons, Ltd, 2004. (cited in page(s) v, 1)

[51] J. Dongarra, V. Eijkhout, and H. Van der Vorst. An Iterative Solver Benchmark. Scientific Programming, 9(4):223–231, 2001. (cited in page(s) 65, 74, 96)

[52] J. Dongarra and F. Sullivan. Guest Editors Introduction to the top 10 algorithms. Computing in Science Engineering, 2(1):22–23, 2000. (cited in page(s) 15, 19)

[53] I. S. Duff and J. K. Reid. The Multifrontal Solution of Indefinite Sparse Symmetric Linear. ACM Trans. Math. Softw., 9(3):302–325, 1983. (cited in page(s) vii, 3, 16, 18)

[54] L. F. R. Espath, A. F. Sarmiento, L. Dalcin, and V.M. Calo. On the thermodynamics of the Swift-Hohenberg theory. Continuum Mechanics and Thermodynamics, pages 1–11, 2017. (cited in page(s) 46)

[55] L. F. R. Espath, A. F. Sarmiento, P. Vignal, B. O. N. Varga, A. M. A. Côrtes, L. Dalcin, and V.M. Calo. Energy exchange analysis in droplet dynamics via the Navier-Stokes-Cahn-Hilliard model. 797:389–430, 2016. (cited in page(s) 46)

[56] N. J. H. et al. (eds.). The Princeton Companion to Applied Mathematics. Princeton University Press, 2015. (cited in page(s) 15, 19)

[57] J. A. Evans, Y. Bazilevs, I. Babuška, and TJR. Hughes. n-widths, sup–infs, and optimality ratios for the k-version of the isogeometric finite element method. Computer Methods in Applied Mechanics and Engineering, 198(21–26):1726 – 1741, 2009. Advances in Simulation-Based Engineering Sciences – Honoring J. Tinsley Oden. (cited in page(s) 7)

[58] J. A. Evans and TJR. Hughes. Isogeometric divergence-conforming B-splines for the Darcy-Stokes-Brinkman equations. Mathematical Models and Methods in Applied Sciences, 23(04):671–741, 2013. (cited in page(s) 78, 89)

[59] J. A. Evans and TJR. Hughes. Isogeometric divergence-conforming B-splines for the steady Navier-Stokes equations. Mathematical Models and Methods in Applied Sciences, 23(08):1421–1478, 2013. (cited in page(s) 78)

[60] J. A. Evans and TJR. Hughes. Isogeometric divergence-conforming B-splines for the unsteady Navier-Stokes equations. Journal of Computational Physics, 241:141–167, 2013. (cited in page(s) vi, 2, 78)

[61] P. Fischer, M. Klassen, J. Mergheim, P. Steinmann, and R. Müller. Isogeometric analysis of 2D gradient elasticity. Computational Mechanics, 47(3):325–334, 2011. (cited in page(s) vi, 2)

[62] R. Fletcher. Conjugate gradient methods for indefinite systems, pages 73–89. Springer Berlin Heidelberg, 1976. (cited in page(s) 19)

[63] L. P. Franca, S. L. Frey, and TJR. Hughes. Stabilized finite element methods: I. Application to the advective-diffusive model. Computer Methods in Applied Mechanics and Engineering, 95(2):253 – 276, 1992. (cited in page(s) v, 1)

[64] K. Gahalaut and S. Tomar. Condition number estimates for matrices arising in the isogeometric discretizations. RICAM report, 23(2012):1–38, 2012. (cited in page(s) 87)

[65] D. Garcia, M. Bartoň, and D. Pardo. Optimally refined isogeometric analysis. Procedia Computer Science, 108:808–817, 2017. International Conference on Computational Science, {ICCS} 2017, 12-14 June 2017, Zurich, Switzerland. (cited in page(s) 5, 22, 83)

[66] D. Garcia, D. Pardo, L. Dalcin, and V.M. Calo. Refined Isogeometric Analysis for a Preconditioned Conjugate Gradient Solver. Computer Methods in Applied Mechanics and Engineering, 2018. (in press). (cited in page(s) 5, 22, 83)

[67] D. Garcia, D. Pardo, L. Dalcin, M. Paszyński, N. Collier, and V.M. Calo. The value of continuity: Refined isogeometric analysis and fast direct solvers. Computer Methods in Applied Mechanics and Engineering, 316:586–605, 2017. Special Issue on Isogeometric Analysis: Progress and Challenges. (cited in page(s) 5, 22, 29, 32, 83, 95)

[68] J. E. Gentle. Iterative Methods for Sparse Linear Systems. Springer-Verlag, Berlin, Ge, 1998. (cited in page(s) 40)

[69] A. George. Nested Dissection of a Regular Finite Element Mesh. SIAM Journal on Numerical Analysis, 10(2):345–363, 1973. (cited in page(s) 16)

[70] H. Gómez, V.M. Calo, Y. Bazilevs, and TJR. Hughes. Isogeometric analysis of the Cahn-Hilliard phase-field model. Computer Methods in Applied Mechanics and Engineering, 197(49–50):4333 – 4352, 2008. (cited in page(s) 2)

[71] H. Gómez, TJR. Hughes, X. Nogueira, and V.M. Calo. Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations. Computer Methods in Applied Mechanics and Engineering, 199(25-28):1828 – 1840, 2010. (cited in page(s) vi, 2)

[72] J. Gopalakrishnan and J. Guzmán. Symmetric Nonconforming Mixed Finite Elements for Linear Elasticity. SIAM Journal on Numerical Analysis, 49(4):1504–1520, 2011. (cited in page(s) v, 1)

[73] A. Greenbaum. Iterative Methods for Solving Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997. (cited in page(s) 19)

[74] M. H. Gutknecht. Block Krylov space methods for linear systems with multiple right-hand sides: an introduction. 2006. (cited in page(s) 19)

[75] M. H. Gutknecht. A Brief Introduction to Krylov Space Methods for Solving Linear Systems, pages 53–62. Springer Berlin Heidelberg, 2007. (cited in page(s) 19, 20)

[76] M. R. Hestenes and E. Stiefel. Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards, 49(6):409–436, 1952. (cited in page(s) 19, 39)

[77] S. S. Hossain, S. F. A. Hossainy, Y. Bazilevs, V.M. Calo, and TJR. Hughes. Mathematical modeling of coupled drug and drug-encapsulated nanoparticle transport in patient-specific coronary artery walls. Computational Mechanics, 49(2):213–242, 2011. (cited in page(s) vii, 3)

[78] TJR. Hughes. The Finite Element Method: Linear Static and Dynamic Finite Element Analysis. Dover Publications, New York, Mi, 2000. (cited in page(s) v, 1)

[79] TJR. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering, 194(39–41):4135 – 4195, 2005. (cited in page(s) vi, vii, 2, 3, 7, 9, 12)

[80] TJR. Hughes, I. Levit, and J. Winget. An element-by-element solution algorithm for problems of structural and solid mechanics. Computer Methods in Applied Mechanics and Engineering, 36(2):241–254, 1983. (cited in page(s) viii)

[81] B. M. Irons. A frontal solution program for finite element analysis. International Journal for Numerical Methods in Engineering, 2:5–32, 1970. (cited in page(s) 16, 18)

[82] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, and TJR. Hughes. An immersogeometric variational framework for fluid-structure interaction: Application to bioprosthetic heart valves. Computer Methods in Applied Mechanics and Engineering, 284:1005 – 1053, 2015. (cited in page(s) vi, 2)

[83] G. Karypis and V. Kumar. A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing, 20(1):359–392, 1999. (cited in page(s) 46)

[84] Karypis Laboratory. METIS, http://glaros.dtc.umn.edu/gkhome/metis/metis/overview, 2016. (cited in page(s) 46, 86)

[85] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with Kirchhoff–Love elements. Computer Methods in Applied Mechanics and Engineering, 198(49):3902–3914, 2009. (cited in page(s) vi, 2)

[86] C. Lanczos. Solution of systems of linear equations by minimized iterations. J. Res. Natl. Bur. Stand, 49:33–53, 1952. (cited in page(s) 19)

[87] S. Lipton, J. A. Evans, Y. Bazilevs, T. Elguedj, and TJR. Hughes. Robustness of isogeometric structural discretizations under severe mesh distortion. Computer Methods in Applied Mechanics and Engineering, 199(5-8):357 – 373, 2010. (cited in page(s) vi, vii, 3)

[88] J. Lu and X. Zhou. Cylindrical element: Isogeometric model of continuum rod. Computer Methods in Applied Mechanics and Engineering, 200(1):233–241, 2011. (cited in page(s) vi, 2)

[89] J. Mandel. Balancing domain decomposition. Communications in Numerical Methods in Engineering, 9(3):233–241, 1993. (cited in page(s) 97)

[90] J. Mandel and C. R. Dohrmann. Convergence of a balancing domain decomposition by constraints and energy minimization. Numerical Linear Algebra with Applications, 10(7):639–659, 2003. (cited in page(s) 97)

[91] N. D. Manh, A. Evgrafov, A. R. Gersborg, and J. Gravesen. Isogeometric shape optimization of vibrating membranes. Computer Methods in Applied Mechanics and Engineering, 200(13):1343–1353, 2011. (cited in page(s) vii, 3)

[92] P. McHugh, D. Knoll, and D. Keyes. Application of Newton-Krylov-Schwarz algorithm to low-mach-number compressible combustion (tn). AIAA journal, 36(2):290–292, 1998. (cited in page(s) 19)

[93] G. A. Meurant. Computer Solution of Large Linear Systems, volume 28 of Studies in Mathematics and Its Applications. North-Holland, Amsterdam, 1999. (cited in page(s) 19)

[94] C. B. Moler. Iterative Refinement in Floating Point. J. ACM, 14(2):316–321, 1967. (cited in page(s) 87)

[95] Y. G. Motlagh, H. T. Ahn, TJR. Hughes, and V.M. Calo. Simulation of laminar and turbulent concentric pipe flows with the isogeometric variational multiscale method. Computers & Fluids, 71:146 – 155, 2013. (cited in page(s) vi, 2)

[96] V. P. Nguyen, C. Anitescu, S. P. Bordas, and T. Rabczuk. Isogeometric analysis: An overview and computer implementation aspects. Mathematics and Computers in Simulation, 117:89 – 116, 2015. (cited in page(s) xiii, 2, 7)

[97] P. N. Nielsen, A. R. Gersborg, J. Gravesen, and N. L. Pedersen. Discretizations in isogeometric analysis of Navier-Stokes flow. Computer Methods in Applied Mechanics and Engineering, 200(45-46):3242 – 3253, 2011. (cited in page(s) vi, 2)

[98] J. Nitsche. Über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind. In Abhandlungen aus dem mathematischen Seminar der Universität Hamburg, volume 36, pages 9–15. Springer, 1971. (cited in page(s) 80)

[99] D. Pardo, J. Álvarez Aramberri, M. Paszyński, L. Dalcin, and V.M. Calo. Impact of element-level static condensation on iterative solver performance. Computers & Mathematics with Applications, 70(10):2331 – 2341, 2015. (cited in page(s) viii, ix, 4, 43)

[100] M. Paszyński. Fast solvers for mesh based computations. Taylor & Francis, CRC Press, 2015. (cited in page(s) 18, 25)

[101] C. G. Petra, O. Schenk, and M. Anitescu. Real-time stochastic optimization of complex energy systems on high-performance computers. IEEE Computing in Science & Engineering, 16(5):32–42, 2014. (cited in page(s) 49)

[102] C. G. Petra, O. Schenk, M. Lubin, and K. Gärtner. An augmented incomplete factorization approach for computing the schur complement in stochastic optimization. SIAM Journal on Scientific Computing, 36(2):C139–C162, 2014. (cited in page(s) 49)

[103] L. Piegl and W. Tiller. The NURBS book. Springer Science & Business Media, 2012. (cited in page(s) 7, 9, 13)

[104] V. Puzyrev, Q. Deng, and V.M. Calo. Spectral approximation properties of isogeometric analysis with variable continuity. Computer Methods in Applied Mechanics and Engineering, 334:22–39, 2018. (cited in page(s) 96)

[105] X. Qian. Full analytical sensitivities in NURBS based isogeometric shape optimization. Computer Methods in Applied Mechanics and Engineering, 199(29):2059–2071, 2010. (cited in page(s) vii, 3)

[106] X. Qian and O. Sigmund. Isogeometric shape optimization of photonic crystals via coons patches. Computer Methods in Applied Mechanics and Engineering, 200(25):2237–2255, 2011. (cited in page(s) vii, 3)

[107] P.-A. Raviart and J.-M. Thomas. A mixed finite element method for 2-nd order elliptic problems. In Mathematical aspects of finite element methods, pages 292–315. Springer, 1977. (cited in page(s) 79)

[108] Y. Saad. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. (cited in page(s) 19, 20, 21, 39, 43)

[109] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput., 7(3):856–869, 1986. (cited in page(s) 19)

[110] A. Sarmiento, A. Côrtes, D. Garcia, L. Dalcin, N. Collier, and V.M. Calo. PetIGA-MF: A multi-field high-performance toolbox for structure-preserving B-splines spaces. Journal of Computational Science, 18:117–131, 2017. (cited in page(s) 46, 78, 86)

[111] A. Sarmiento, D. Garcia, L. Dalcin, N. Collier, and V. Calo. Micropolar Fluids Using B-spline Divergence Conforming Spaces. Procedia Computer Science, 29:991 – 1001, 2014. (cited in page(s) 46)

[112] O. Schenk, K. Gärtner, and W. Fichtner. Efficient sparse lu factorization with left-right looking strategy on shared memory multiprocessors. BIT Numerical Mathematics, 40(1):158–176, 2000. (cited in page(s) 49)

[113] V. Simoncini and D. B. Szyld. Recent computational developments in Krylov subspace methods for linear systems. Numerical Linear Algebra with Applications, 14(1):1–59, 2007. (cited in page(s) 19)

[114] A. Tagliabue, L. Dedé, and A. Quarteroni. Isogeometric Analysis and error estimates for high order partial differential equations in fluid dynamics. Computers & Fluids, 102:277 – 303, 2014. (cited in page(s) vi, 2)

[115] C. V. Verhoosel, M. A. Scott, TJR. Hughes, and R. De Borst. An isogeometric analysis approach to gradient damage models. International Journal for Numerical Methods in Engineering, 86(1):115–134, 2011. (cited in page(s) vi, 2)

[116] P. Vignal, L. Dalcin, D. L. Brown, N. Collier, and V.M. Calo. An energy-stable convex splitting for the phase-field crystal equation. Computers & Structures, 158:355 – 368, 2015. (cited in page(s) 2, 46)

[117] P. Vignal, L. Dalcin, N. Collier, and V.M. Calo. Modeling Phase-transitions Using a High-performance, Isogeometric Analysis Framework. Procedia Computer Science, 29:980 – 990, 2014. (cited in page(s) 2)

[118] P. Vignal, A. Sarmiento, A. M. Côrtes, L. Dalcin, and V.M. Calo. Coupling Navier-Stokes and Cahn-Hilliard equations in a two-dimensional annular flow configuration. Procedia Computer Science, 51:934 – 943, 2015. (cited in page(s) 2)

[119] P. A. Vignal, N. Collier, and V.M. Calo. Phase Field Modeling Using PetIGA. Procedia Computer Science, 18:1614 – 1623, 2013. (cited in page(s) 2, 46)

[120] W. A. Wall, M. A. Frenzel, and C. Cyron. Isogeometric structural shape optimization. Computer methods in applied mechanics and engineering, 197(33):2976–2988, 2008. (cited in page(s) vii, 3)

[121] J. H. Wilkinson. Rounding Errors in Algebraic Processes. Dover Publications, Incorporated, 1994. (cited in page(s) 87)

[122] E. L. Wilson. The static condensation algorithm. International Journal for Numerical Methods in Engineering, 8(1):198–203, 1974. (cited in page(s) 32)

[123] Y. Zhang, Y. Bazilevs, S. Goswami, C. L. Bajaj, and TJR. Hughes. Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Computer Methods in Applied Mechanics and Engineering, 196(29-30):2943 – 2959, 2007. (cited in page(s) vi, vii, 2, 3)

[124] O. Zienkiewicz and R. Taylor. The Finite Element Method for Solid and Structural Mechanics. Elsevier Science, 2013. (cited in page(s) v, 1)

[125] O. Zienkiewicz, R. Taylor, and P. Nithiarasu. The Finite Element Method for Fluid Dynamics. Elsevier Science, 2013. (cited in page(s) v, 1)

[126] S. Zlotnik, P. Díez, M. Fernández, and J. Vergés. Numerical modelling of tectonic plates subduction using X-FEM. Computer Methods in Applied Mechanics and Engineering, 196(41):4283 – 4293, 2007. (cited in page(s) v, 1)